Universidad de las Ciencias Informáticas

Facultad 2

Título: SEPI: Sistema para evaluar proyectos y establecer un orden de prioridad que ayude a la toma de decisiones.

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

Autor: Leandro José Rodríguez Hernández

Tutores: MSc. Karina Sánchez Tamayo

Ing. Rammel Maestre Sánchez

Junio, 2012

Pensamiento

En dos palabras puedo resumir cuanto he aprendido acerca de la vida: Sigue adelante.

Robert Lee Frost.

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Leandro José Rodriguez Hernández

_____

Firma del autor

MSc. Karina Sánchez Tamayo

_____

Firma del tutor

Ing. Rammel Maestre Sánchez

_____

Firma del tutor

Agradecimientos

A mis padres, por deberles todo.

A Dios.

A mi novia Aliuska.

A mis tutores, los profesores Karina y Rammel.

A la profesora Yadira Benavides.

A mis amigos y familiares.

Dedicatoria

A mis padres.

Resumen

En el presente trabajo se describe la construcción de una solución informática que permite la aplicación de MEPROI, método para evaluar proyectos y establecer un orden de prioridad que ayude a la toma de decisiones.

La solución construida en un ambiente web, bajo el marco de trabajo Grails y PostgreSQL como gestor de base de datos, siguiendo la metodología ágil XP para su desarrollo, sin la presencia de otros sistemas precedentes, soporta elementos determinantes de MEPROI como son la consulta a expertos y el procedimiento estadístico en que se apoya, facilitando a su vez la disponibilidad de los resultados.

Con el fin de lograr este propósito, se realizó un estudio del método MEPROI y las etapas que lo componen. Se definen los requerimientos que debe cumplir la aplicación, modelándose además todo el proceso de construcción de la misma hasta la etapa de pruebas. Durante esta etapa fueron realizadas pruebas de unidad y aceptación con resultados satisfactorios.

Palabras clave:

Herramientas de apoyo a la toma de decisiones, índice de prioridad, métodos de evaluación de proyectos.

Índice

Índice de Figuras

Índice de Tablas

Introducción

La evaluación de proyectos es una actividad que existe desde que el hombre tuvo que comenzar a evaluar alternativas en función de sus necesidades o beneficios. Desde muy temprano el ser humano tuvo que enfrentarse a la necesidad de establecer un orden de prioridad en las tareas que tenía que acometer. Aprender a realizar un balance entre el costo de realizar la tarea y el beneficio que podía acarrearle.

Primero, esta evaluación se realizaba de forma muy empírica e intuitiva, luego, con el paso del tiempo, el desarrollo de la sociedad y de las relaciones económico-mercantiles, fueron surgiendo nuevas tareas o proyectos de mayor complejidad. También fueron necesarios métodos de evaluación de proyectos más complejos y eficientes. Esto fue posible gracias al desarrollo de las matemáticas, y más recientemente al desarrollo de medios de cómputo capaces de agilizar la aplicación de estos métodos.

Para tomar una decisión sobre un proyecto es necesario que este sea sometido al análisis multidisciplinario de diferentes especialistas. Una decisión siempre debe estar basada en el análisis de un sin número de antecedentes o la aplicación de una metodología que abarque la consideración de todos los factores que participan y afectan al proyecto.

“Por estas razones, la toma de decisiones acerca de invertir en determinado proyecto siempre debe recaer no en una sola persona ni en el análisis de datos parciales, sino en grupos multidisciplinarios que cuenten con la mayor cantidad de información posible. A toda actividad encaminada a tomar una decisión de inversión sobre un proyecto se le llama evaluación de proyectos” (1)

La evaluación de proyectos consiste en comparar los costos con los beneficios que estos generan, para así decidir sobre la conveniencia de llevarlos a cabo. Esta pretende abordar el problema de la asignación de recursos en forma explícita, recomendando a través de distintas técnicas que una determinada iniciativa se lleva adelante por sobre otras alternativas de proyectos.

Para evaluar un proyecto se tienen en cuenta un amplio espectro de indicadores, indicadores financieros, económicos, sociales, técnicos, entre otros. Las evaluaciones financiera, económica y social se efectúan conjuntamente con la que se podría llamar evaluación técnica del proyecto, que consiste en cerciorarse de la factibilidad técnica del mismo. Así mismo, la

evaluación económica presupone una adecuada formulación y evaluación administrativa, como también una adecuada formulación y evaluación institucional y legal.

“La evaluación de proyectos ofrece significativas ventajas al sistema nacional de ciencia e innovación tecnológica, pues se ha convertido en un elemento fundamental para su organización, priorizar los que presentan mayores posibilidades de éxito, reducir el tiempo entre la obtención de los resultados y su introducción en la práctica social, impulsar el desarrollo de los programas priorizados, favorecer el ambiente creativo de los investigadores, garantizar un mejor empleo de los recursos y estimular los mejores esfuerzos y resultados.” (2)

“Un proyecto surgido en medio de incertidumbre e indefiniciones, donde no se usan métricas para guiar y mejorar su desempeño y donde los directivos no tengan conocimiento de su desarrollo, ni medios de obtenerlo para influir de alguna manera, es un proyecto muy propenso al fracaso. La ausencia de métodos y procedimientos que permitan evaluar elementos importantes del proceso de producción es una variable importante a tener en cuenta en el trabajo de la organización para mejorar este proceso.” (3)

La Universidad de las Ciencias Informáticas (UCI), como entidad desarrolladora de proyectos informáticos, tiene la necesidad de evaluar un número elevado de proyectos de software, con el objetivo de establecer una prioridad en su ejecución con un análisis de factibilidad previo.

En los primeros años de la Universidad, se hacía muy complejo este proceso de evaluar los proyectos y tomar decisiones. Esto era debido a que no se manejaba toda la información necesaria y no se aplicaban métodos con bases científicas para evaluar el comportamiento de indicadores que influyen en el desarrollo del proceso y poder determinar un índice de prioridad para cada proyecto. Surgían problemas como: atrasos en los plazos de ejecución, no contar con los recursos humanos y materiales necesarios para asumir el proyecto, no tener clara la magnitud del proyecto desde su fase inicial.

En esos años, no se tenía en cuenta un método formal, con basamentos científicos para realizar la comparación entre los diferentes proyectos. Cuando una propuesta era presentada a la UCI, se le aceptaba y se encargaba su ejecución a un grupo de desarrollo. La carencia de un método de evaluación de proyectos, hacía que en ocasiones se tomaran decisiones erróneas, guiadas solo por la intuición de las personas designadas para tal función. Se necesitaba una vía para poder discernir qué proyecto resulta más favorable que otro, pues en caso de no poder

satisfacer todas las propuestas a la vez, se debían enfocar los recursos y el personal hacia los de mayor prioridad.

Por causa de la gran cantidad de proyectos tanto nacionales, como foráneos que maneja la UCI, se hacía necesario encontrar formas más favorables de valorarlos, haciendo uso para esto de técnicas de evaluación de proyectos existentes, que permiten establecer un orden de prioridad para su ejecución.

Como una solución a este problema surge MEPROI, un método para evaluar proyectos incluyendo una propuesta de indicadores o criterios acordes a los proyectos de la Universidad y establecer un orden de prioridad que ayude a la toma de decisiones. Este método fue resultado de una investigación realizada para la defensa de una tesis de maestría en el año 2010. Es un método basado en un procedimiento matemático-estadístico que teniendo en cuenta los criterios de un grupo de evaluadores respecto a diversas aristas de los proyectos, es capaz de establecer un índice de prioridad para cada una de las propuestas.

La aplicación actual del método luego de tener la información que proporciona cada evaluador se realiza a punta de lápiz, por lo que se pudiera incurrir en errores durante el procesamiento estadístico que determina la confiabilidad del método y luego en el cálculo de los índices de prioridad general y por grupos de criterios, resultados finales que éste brinda.

Se incurre en demoras de tiempo hasta obtener un resultado pues la interacción con los evaluadores es de forma personal por lo que es necesario tener en cuenta la disponibilidad de la persona, tanto física como intelectualmente.

De acuerdo a la situación planteada anteriormente surge el siguiente **problema a resolver**:
La aplicación actual del método MEPROI para evaluar proyectos y establecer un orden de prioridad que ayude a la toma de decisiones dificulta la recolección de los datos, la exactitud en el procesamiento estadístico y la disponibilidad de los resultados.

Por tanto se define como el **objeto de estudio:** Herramientas de apoyo a la toma de decisiones.

El **objetivo general** es: Desarrollar un sistema informático que sustente la aplicación del método MEPROI para evaluar proyectos y establecer un orden de prioridad que ayude a la

toma de decisiones que facilite la recolección de los datos, exactitud en el procesamiento estadístico y disponibilidad de los resultados.

Delimitando así el **campo de acción,** siendo este, el proceso de aplicación del método MEPROI para evaluar proyectos y establecer un orden de prioridad que ayude a la toma de decisiones.

Los **objetivos específicos** que se derivan son los siguientes:

1. Elaborar el marco teórico.
2. Definir el entorno de desarrollo para la solución informática.
3. Diseñar la solución informática para evaluar proyectos y establecer un orden de prioridad que ayude a la toma de decisiones.
4. Implementar la solución informática para evaluar proyectos y establecer un orden de prioridad que ayude a la toma de decisiones.
5. Probar la solución informática para evaluar proyectos y establecer un orden de prioridad que ayude a la toma de decisiones.

Este trabajo consta de: introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos. El **primer capítulo** aborda los conceptos relacionados y la fundamentación teórica que en su conjunto ayudan a la comprensión de la solución del problema que se plantea, también se define la metodología de desarrollo de software a utilizar. En el **segundo capítulo** se detallan las herramientas que serán utilizadas para el desarrollo, las diferentes características que va a presentar el sistema, el modelo de datos, una argumentación de la arquitectura y los patrones de diseño a utilizar. El **tercer capítulo** está orientado a la implementación a partir de los artefactos que define la metodología a seguir y las pruebas a realizar al sistema.

Capítulo 1: Fundamentación teórica

Un método empleado desde la perspectiva de priorizar los proyectos es MEPROI, método para evaluar proyectos y establecer un orden de prioridad que ayude a la toma de decisiones.
En el presente capítulo se analizan los aspectos más importantes que servirán como soporte teórico a la construcción de la aplicación, se abordan temas relacionados con la toma de decisiones, se describe el método MEPROI y se define la metodología que guiará el proceso de desarrollo de la solución.

1.1 Marco conceptual

A continuación se detallan los conceptos relacionados con el entorno del problema para un mayor entendimiento del marco teórico elaborado.

Proyecto: Conjunto de actividades relacionadas para lograr un fin específico, con un comienzo y fin claros, sujeto a tres "restricciones" principales: Tiempo, Presupuesto y Alcance. (4)

Gestión de proyectos: También conocida como Gerencia, Dirección o Administración de proyectos es la disciplina de planear, organizar, asegurar y coordinar recursos y personas para cumplir con los Objetivos, Entregables y Criterios de Éxito de los proyectos. (4)

Evaluación de proyectos: es un proceso por el cual se determina el establecimiento de cambios generados por un proyecto a partir de la comparación entre el estado actual y el estado previsto en su planificación. Se intenta conocer qué tanto un proyecto ha logrado cumplir sus objetivos o bien qué tanta capacidad poseería para cumplirlos.

Toma de decisiones: proceso de selección entre cursos alternativos de acción, basado en un conjunto de criterios, para alcanzar uno o más objetivos. (3)

1.2 Proceso de toma de decisiones

La toma de decisiones constituye un factor determinante en el éxito o el fracaso del desempeño de las organizaciones e instituciones.

La separación del proceso en etapas puede ser tan resumida o tan extensa como se desee, pero se pueden identificar principalmente las siguientes etapas:

Tabla 1. Etapas del proceso de toma de decisiones

Etapas	**Descripción**
Identificar y analizar el problema	Esta etapa consiste en encontrar el problema y reconocer que se debe tomar una decisión para llegar a la solución de este.
Identificar los criterios de decisión y ponderarlos	Consiste en identificar aquellos aspectos que son relevantes al momento de tomar la decisión, es decir aquellas pautas de las cuales depende la decisión que se tome. La ponderación, es asignar un valor relativo a la importancia que tiene cada criterio en la decisión que se tome, ya que todos son importantes pero no de igual forma.
Definir la prioridad para atender el problema	La definición de la prioridad se basa en el impacto y en la urgencia que se tiene para atender y resolver el problema.
Generar las opciones de solución	Consiste en desarrollar distintas posibles soluciones al problema. Si bien no resulta posible en la mayoría de los casos conocer todos los posibles caminos que se pueden tomar para solucionar el problema, cuantas más opciones se tengan va ser mucho más probable encontrar una que resulte satisfactoria.
Evaluar las opciones	Consiste en hacer un estudio detallado de cada una de las posibles soluciones que se generaron para el problema, es decir mirar sus ventajas y desventajas, de forma individual con respecto a los criterios de decisión, y una con respecto a la otra, asignándoles un valor ponderado.
Elección de la mejor opción	En esta etapa del proceso es importante el análisis crítico como cualidad del tomador de decisiones. Se escoge la opción que según la evaluación va a obtener mejores resultados para el problema.
Aplicación de la decisión	Poner en marcha la decisión tomada para así poder evaluar si la decisión fue o no acertada.
Evaluación de los resultados	Después de poner en marcha la decisión es necesario evaluar si se solucionó o no el problema, es decir si la decisión está teniendo el resultado esperado o no.

El método de ayuda a la toma de decisiones bajo el que se sustenta el desarrollo de este trabajo sigue las etapas abordadas hasta la de *Evaluar las opciones* dando como resultado las opciones priorizadas de acuerdo a los criterios definidos para la evaluación.

1.3 MEPROI

El ***M***étodo de ***E***valuación de ***PRO***yectos, la ***I*** por ser aplicado a proyectos Informáticos (***MEPROI***) fue el resultado exitoso de una investigación realizada para optar por el título de máster en la maestría de Gestión de proyectos informáticos en el año 2010 de la tutora del presente trabajo de diploma.

Surge como respuesta a la necesidad de contar con un método científicamente fundamentado para establecer prioridades en la ejecución de los proyectos informáticos acorde a los intereses de la organización.

El método se sustenta en las técnicas de decisiones multicriterios, aplicando procedimientos estadísticos y matemáticos sobre los criterios de un grupo de personas expertas en el entorno de los proyectos a evaluar.

1.3.1 Objetivo y alcance

El método MEPROI se aplica a cualquier tipo de proyecto, preferiblemente contratado, es decir que sean factibles en su realización (estudio de factibilidad previo), llevando a cabo una evaluación para cada proyecto con la participación de expertos que arroje como resultado un índice de prioridad para cada proyecto y una propuesta para el orden de ejecución según los grupos de criterios definidos.

1.3.2 Artefactos de entrada al método

Para llevar a cabo el método es necesario información referente a los antecedentes del proyecto, alcance, problema a resolver, beneficiarios, montos por cada uno de los subproyectos de cada uno de los proyectos que serán sometidos a la evaluación.

1.3.3 Actores que intervienen

El método define tres actores a participar:

Analista: Dirige todo el proceso de evaluación, define los criterios a utilizar, selecciona los evaluadores y aplica el método de evaluación para establecer el orden de los proyectos.

Evaluador: Expresa la calidad del proyecto de acuerdo con sus conocimientos sobre la temática que trata y califica los criterios empleados según la escala pre-establecida.

Decisor: Analiza los resultados obtenidos de la aplicación del método y aprueba el proyecto de acuerdo con la importancia de los criterios utilizados.

1.3.4 Etapas del método

1.3.4.1 Etapa 1: Determinar el índice de prioridad para los proyectos

1. *Definir los criterios de evaluación por grupos*

Los criterios para la evaluación de los proyectos se definen por el Analista a partir de los intereses de la organización, pueden ser modificados de acuerdo con las características del proyecto o los intereses de la organización.

El método propone un total de 27 criterios a utilizar para la evaluación organizados en los grupos de criterios siguientes: méritos científicos, económicos, comercial, impacto e informáticos.

No.	Criterios
	Méritos científicos
1	Valor científico del proyecto
2	Calidad de la organización del proyecto
3	Planificación realizada para su ejecución
4	Idoneidad científica, profesional y gerencial del líder de proyecto
5	Solvencia científica y profesional del equipo de proyecto
	Económicos
6	Cantidad de instituciones participantes y calidad de su infraestructura y servicios
7	Medios materiales disponibles y solicitados
8	Presupuesto solicitado para su ejecución
9	Presupuesto solicitado para su introducción
10	Ganancias esperadas
	Comerciales
11	Satisfacción de los requerimientos del cliente
12	Ventajas del producto, proceso o servicios que se desarrollan
13	Atractivos del mercado al que se puede acceder
14	Nivel de competencia existente
15	Requerimientos para la introducción en el mercado
16	Posibilidad de acceder a nuevo mercado
	Impacto
17	Impacto social
18	Impacto medioambiental
19	Impacto en el territorio donde se introduce
20	Impacto político

	Informáticos
21	Impacto de las regulaciones legales en el lugar de aplicación
22	Tecnología disponible para ejecutar el proyecto
23	Existencia de componentes reutilizables
24	Impacto de los riesgos identificados
25	Impacto de variaciones de alcance
26	Complejidad del proyecto y del propio cliente
27	Dependencias con otros sistemas e integración

Tabla 2. Criterios propuestos para la evaluación

2. *Seleccionar los expertos*

De acuerdo con el enfoque dado a la gestión de proyectos en el PMBOK[1] las entidades que se dedican a desarrollar proyectos deben especializar un área de la empresa como oficina de gestión de proyectos (PMO en sus siglas en inglés), por lo que deben contar con los expertos suficientes para hacer estas evaluaciones, deben siempre ser un número igual o mayor a ocho expertos de acuerdo con los criterios estadísticos utilizados para procesar la información obtenida.

Los posibles expertos deben ser sometidos a una selección previa basada en la técnica usada en el método Delphi para la selección de los expertos. (5)

3. *Elaborar guía de evaluación a los expertos*

Se elabora la guía de evaluación para ser entregada a los expertos con un plazo fijo para terminar su trabajo, la guía no es más que la relación de los criterios con el peso o valor relativo por asignar a cada uno.

4. *Asignar un valor relativo a cada criterio*

Se entenderá por valor relativo a la importancia que se le atribuye a la evaluación de ese criterio para el proyecto con respecto al resto de los criterios. Para ello se le pide a cada experto que emita su opinión acerca de la importancia (en base a 100) que tiene cada indicador con relación a los demás para el proyecto a evaluar. La sumatoria de estos valores relativos debe ser igual a 100.

[1] Guía de los Fundamentos de la Dirección de Proyectos, referencia del Instituto de Gestión de Proyecto (**P**roject **M**anagement **I**nstitute)

C/E	E1	E2	E3	E4	E5	E6	E7	E8	$\sum$ E	EP
C_1	V_{11}	V_{12}	...	...	...	...	V_{17}	V_{18}	$\sum V_1$	$\frac{\sum V_1}{E}$
C_2	V_{21}	V_{22}	...	...	...	...	V_{27}	V_{28}	$\sum V_2$	$\frac{\sum V_2}{E}$
...	...	...	...	...	...	...	...	...	...	...
C_n	V_{n1}	V_{n2}	...	...	...	...	V_{n7}	V_{n8}	$\sum V_n$	$\frac{\sum V_n}{E}$
									$\sum\sum V$	$\sum VP$

Figura 1. Matriz de evaluación de expertos

C_i: criterio

E_j: experto

V_{ij}: valor asignado por el experto *i* al criterio *j*

EP: promedio de los valores por cada criterio

VP: sumatoria de los valores promedio.

Se le puede asignar un peso relativo a cada grupo de criterios según la evaluación.

5. *Verificar concordancia en el trabajo de los expertos*

Es importante verificar el consenso de los expertos para tener confianza en los datos y continuar con el proceso de evaluación, para ello se utiliza el coeficiente de concordancia de Kendall (W)[2] con el uso de ligas pues un experto proporciona el mismo peso para diferentes criterios y se utiliza el estadígrafo Chi-Cuadrado para llegar a conclusiones. (6)

Si no existe concordancia entre los expertos entonces se Elimina el experto discordante teniendo en cuenta siempre que el número de expertos ≥ 8 y se verifica nuevamente la concordancia entre los expertos.

6. *Eliminar el experto discordante*

[2] Los valores del coeficiente deben oscilar entre 0 y 1 (0 < W < 1), si W alcanza el valor uno (W = 1) entonces existe una concordancia total de criterios, mientras mayor sea el valor de W, es decir, cuanto más se acerque a uno, mayor será la concordancia entre los expertos.

Para eliminar el experto discordante se utilizan expresiones matemáticas que pueden ahorrar el tiempo de repetir nuevamente la consulta a expertos y dan la posibilidad de determinar los expertos que discordan y poder sustituirlos. (7)

Si existe concordancia entre los expertos se define el peso relativo para cada criterio.

7. *Definir el peso relativo de cada criterio*

El peso relativo de cada indicador se define partiendo de la evaluación anterior a través de la siguiente expresión:

$$P = \frac{EP_n}{100} \text{ donde:}$$

P: peso relativo

EP_n: media aritmética de los valores por cada criterio.

C/E	E1	E2	E3	E4	E5	E6	E7	E8	$\sum$ E	EP	P
C_1	V_{11}	V_{12}	...	...	...	...	V_{17}	V_{18}	$\sum V_1$	$\frac{\sum V_1}{E}$	$\frac{EP_1}{100}$
C_2	V_{21}	V_{22}	...	...	...	...	V_{27}	V_{28}	$\sum V_2$	$\frac{\sum V_2}{E}$	$\frac{EP_2}{100}$
...	...	...	...	...	...	...	...	...	...	...	...
C_n	V_{n1}	V_{n2}	...	...	...	...	V_{n7}	V_{n8}	$\sum V_n$	$\frac{\sum V_n}{E}$	$\frac{EP_n}{100}$
									$\sum\sum V$	$\sum VP$	$\sum P$

Figura 2. Determinación del peso de los criterios en el proyecto

***C**$_i$*: criterio

***E**$_j$*: experto

***V**$_{ij}$**:** valor asignado por el experto *i* al criterio *j*

EP: promedio de los valores por cada criterio

8. *Elaborar guía de evaluación a los expertos. Ídem al 2*
9. *Calificar los criterios*

El paso de calificación de cada indicador consiste en estudiar y evaluar el comportamiento de cada uno de los criterios en el proyecto, en el que se les pide a los expertos que califiquen cada uno en una escala de 1-5 según su opinión y conocimiento acerca del comportamiento que tiene este en el proyecto.

10. Determinar el índice de prioridad.

El índice de prioridad general puede definirse a partir de la calificación dada y el peso relativo definido.

Según los resultados obtenidos se acotan los valores del índice de prioridad:

$IP >$ 0,7 - Prioridad alta

0,7 $> IP >$ 0,5 - Prioridad media

0,5 $> IP >$ 0,3 - Prioridad baja

1.3.4.2 Etapa 2: Determinar el orden de prioridad por grupos de criterios

Con la aplicación del método a todos los proyectos en su primera etapa se puede obtener la información necesaria para llevar a cabo la segunda etapa.

1. *Construir la matriz de prioridad*

Para construir la matriz de prioridad se utiliza la matriz para la calificación de los criterios en el proyecto como se describe a continuación:

P/G	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo n	IP
P_1	IP_{MC1}	IP_{EC1}	IP_{CM1}	IP_{IMP1}	IP_{INF1}	IP_1
P_2	.	.	.	.	.	.
P_n	.	.	.	.	.	.

Figura 3. Matriz proyecto/grupo

siendo:

$IP_{MC1} = \frac{\sum P_{MC} \; x \; c_{MC}}{5}$ (3) ; $IP_1 = IP_{G11} + IP_{G21} + IP_{G31} + IP_{G41} + IP_{Gn1}$ (4) donde:

IP_{G11}, IP_{G21}, IP_{G31}, IP_{G41}, IP_{n1}: Índice de prioridad de los grupos de criterios definidos.

IP_1: Índice de prioridad correspondiente al primer proyecto

2. *Determinar orden de prioridad*

Para determinar el orden de prioridad se utiliza la matriz de prioridad y se construye una matriz Prioridad/Grupo como se muestra a continuación:

P/G	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo n
1	P_2	P_1	P_4	P_3	P_n
2	.	.	.	.	.
$Prioridad_n$	.	.	.	.	.

Figura 4. Matriz Prioridad/Grupo

El proyecto de mayor índice de prioridad para el grupo *Méritos Científicos* será el de mayor prioridad para ese grupo entre los restantes proyectos y así sucesivamente se definen las prioridades para cada uno de los grupos de criterios.

Esta matriz le brindará al decisor la prioridad de cada proyecto en cada grupo de acuerdo a la evaluación realizada previamente. De esta forma el decisor puede valorar cuál de los proyectos influye en mayor o menor grado en un criterio dentro de la organización al ser ejecutado y qué puede esperarse de cada uno.

1.3.4.3 Etapa 3: Análisis de los resultados

En este paso el Decisor analiza los resultados obtenidos a través del método en las etapas anteriores:

- Orden de prioridad general.
- Orden de prioridad por grupos de criterios.

Con estos resultados éste puede tomar una decisión acorde a los intereses de la organización con un mayor nivel de información.

1.4 Metodologías de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software.

En ellas se va indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.

Las técnicas indican cómo debe ser realizada una actividad técnica determinada identificada en la metodología. Combina el empleo de unos modelos o representaciones gráficas junto con el empleo de unos procedimientos detallados. Se debe tener en consideración que una técnica determinada puede ser utilizada en una o más actividades de la metodología de desarrollo de software.

Las metodologías se clasifican en ágiles y tradicionales. Las tradicionales son metodologías que se centran fundamentalmente en el control del proceso, además de ser muy efectivas para proyectos de gran tamaño. Las ágiles son aquellas donde el desarrollo de software se caracteriza por ser incremental, cooperativo, sencillo y adaptable. Estas surgen como una extensión a las metodologías tradicionales para mejorar el desarrollo de sistemas, optimizando las prácticas de desarrollo de software.

1.4.1 Metodologías Tradicionales

Se centran en la definición detallada de los procesos, tareas y herramientas a utilizar, y requiere una extensa documentación, ya que pretende prever todo de antemano. Este tipo de metodologías son más eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear, donde una gran organización es requerida.

1.4.1.1 Rational Unified Process

Una de las metodologías tradicionales más conocidas y utilizadas es la metodología RUP (Rational Unified Process) que divide el desarrollo en 4 fases que definen su ciclo de vida:

- *Inicio*: El objetivo es determinar la visión del proyecto y definir lo que se desea realizar.
- *Elaboración*: Etapa en la que se determina la arquitectura óptima del proyecto.
- *Construcción*: Se obtiene la capacidad operacional inicial.
- *Transmisión*: Obtener el producto acabado y definido.

“Rational Unified Process (RUP) es un marco de proceso integral que ofrece prácticas probadas en la industria de entrega de software e implementación de sistemas y para la gestión eficaz de los proyectos. Es uno de los muchos procesos contenidos dentro de la Biblioteca Proceso Racional, que ofrece mejores directrices prácticas adecuadas para su desarrollo en particular o las necesidades del proyecto.” (8)

RUP propone un gran cúmulo de documentación de acuerdo al tamaño del proyecto y por consecuencia del equipo de desarrollo, es recomendable utilizarla en en el desarrollo de proyectos medianos y grandes.

1.4.2 Metodologías Ágiles

Este tipo de metodologías surgen a finales de la década del 90 como alternativa a los inconvenientes de las metodologías tradicionales.

Las principales ideas de la metodología son:

- Se encarga de valorar al individuo y las iteraciones del equipo más que a las herramientas o los procesos utilizados.
- Se hace mucho más importante crear un producto software que funcione que escribir mucha documentación.
- El cliente está en todo momento colaborando en el proyecto.
- Es más importante la capacidad de respuesta ante un cambio realizado que el seguimiento estricto de un plan.

Existen muchas metodologías ágiles, según un estudio comparativo realizado en el año 2008 (9), destacan las seis que se relacionan a continuación:

1. Agile Project Management.
2. Crystal Methods
3. Dynamic Sytems Development Method.
4. Scrum.
5. Test Driven Development.
6. Extreme Programming.

De ellas Extreme Programming desde 1999 se ha convertido en la punta del iceberg de las metodologías ágiles, es la primera en internet, en adopciones y experiencias y la primera en estudios. (9)

XP consigue abordar la mayoría de los parámetros de calidad e introduce un conjunto de herramientas que implementan técnicas concretas, especialmente en las fases de pruebas.

1.4.2.1 Programación Extrema o XP (eXtreme Programming)

Es una metodología ágil para el desarrollo de software y consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. XP se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. Promueve el trabajo en equipo, preocupándose en todo momento del aprendizaje de los desarrolladores y estableciendo un buen clima de trabajo. Este tipo de método se basa en una realimentación continuada entre el cliente y el equipo de desarrollo con una comunicación fluida entre todos los participantes, también busca simplificar las soluciones implementadas.

“Extreme Programming es exitoso porque hace hincapié en la satisfacción del cliente. En vez de entregar todo lo que pueda desear en una fecha lejana en el futuro este proceso ofrece el software que necesita que usted lo necesite. Extreme Programming permite a los desarrolladores responder con confianza a las necesidades cambiantes de los clientes, incluso al final del ciclo de vida.” (10)

La metodología XP se basa en:

- Pruebas unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que se adelanta en algo hacia el futuro, se puede hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.
- Re-fabricación: se desarrolla en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

Según su creador Kent Beck (11), el ciclo de vida ideal de XP consta de seis fases:

1. ***Exploración***

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2. ***Planificación***

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. En esta etapa también se registra la velocidad del desarrollo en función de la suma de puntos correspondientes a las historias de usuario terminadas por iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto

de historias. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

3. Iteraciones hasta la primera entrega

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

4. Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

5. Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

6. Muerte del Proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios

esperados por el cliente o cuando no hay presupuesto para mantenerlo.

1.4.3 Metodología a utilizar

Para guiar el desarrollo de la solución se decide asumir la metodología XP basado en los aspectos vistos anteriormente y los siguientes:

- El equipo de desarrollo estará integrado por 3 personas en su totalidad.
- El cliente es parte del equipo de desarrollo.
- No existe un contrato tradicional al que darle cumplimiento.
- Es un proyecto considerado pequeño.

El proceso de desarrollo no aborda las fases de Mantenimiento y Muerte del proyecto, pues el objetivo del trabajo es realizar una primera entrega, aún quedan funcionalidades que recomendar para próximas entregas del sistema.

Con el fin de definir cada una de las etapas y sus artefactos en el transcurso del desarrollo se decidió ajustar el ciclo de vida a la propuesta de solución como sigue:

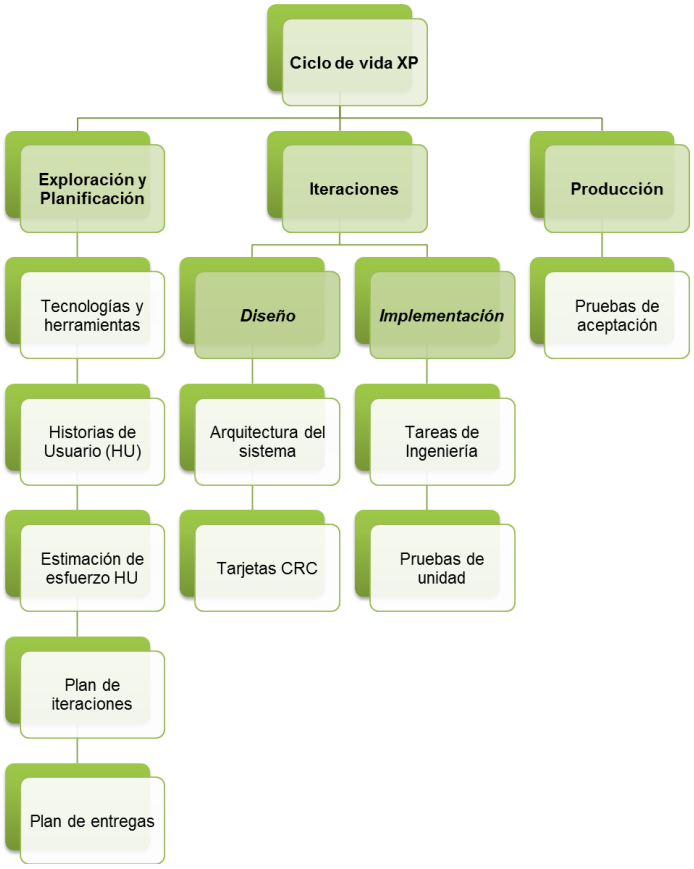


Figura 5. Ciclo de vida XP para guiar la propuesta de solución

1.5 Conclusiones parciales

- El método MEPROI como herramienta de apoyo a la toma de decisiones constituye la base de la propuesta de solución.
- Se define XP como la metodología de desarrollo de software a utilizar.

Capítulo 2: Exploración y Planificación

La tendencia actual en el campo de la producción de software a nivel mundial es obtener productos de software en el menor tiempo posible con elevados estándares de calidad y elaborar la documentación estrictamente necesaria, por lo cual las metodologías de desarrollo ágiles surgen en aras de brindar a los desarrolladores una guía para alcanzar lo anteriormente expuesto.

En el presente capítulo, se aborda el entorno de desarrollo definido, se explican las diferentes características que va a presentar el sistema a implementar, sus funcionalidades siguiendo los aspectos que plantea la metodología ágil XP en sus fases de Exploración y Planificación la cual servirá de guía para tener mayor organización en la distribución del tiempo, actividades y artefactos a desarrollar.

2.1 Tecnologías y herramientas

2.1.1 Marco de Trabajo

La palabra inglesa "framework" define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

“El marco de trabajo o framework, es una estructura conceptual básica usada para resolver un problema complejo. Pueden estar basados en un conjunto de conceptos, modelos, metodologías, componentes, herramientas de administración y de diseño, de manera que facilite la construcción de las aplicaciones manteniendo el orden en las partes que las conforman. Un Framework, simplifica el desarrollo de las mismas mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes, ya que encapsula operaciones complejas en instrucciones sencillas.” (12)

2.1.1.1 Grails

“Grails es una plataforma avanzada e innovadora de código abierto, para el desarrollo web, que ofrece nuevos niveles de productividad de los desarrolladores mediante la aplicación de principios como la Convención sobre configuración. Grails ayuda a los equipos de desarrollo que abrazan las metodologías ágiles a entregar aplicaciones de calidad en menor cantidad de tiempo, y a centrarse en lo realmente importante: la creación de aplicaciones de alta calidad y fáciles de usar. Grails, naturalmente, complementa el desarrollo de aplicaciones Java, ya que

se basa en Spring y sobre la base de Groovy, el principal idioma dinámico para la plataforma Java.” (13)

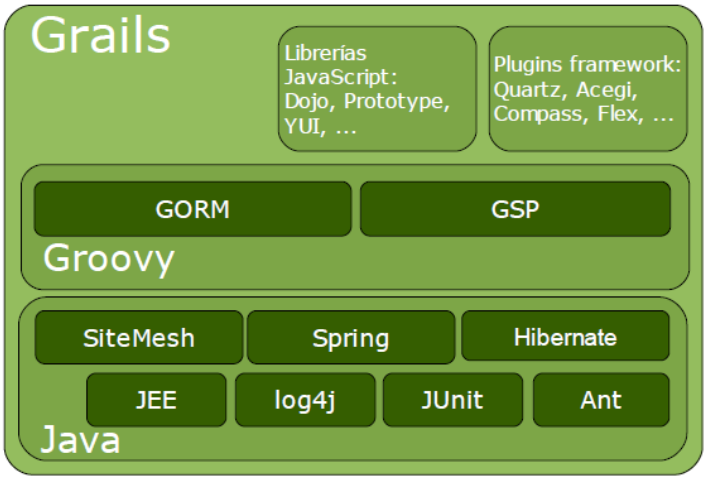


Figura 6. Marco de trabajo Grails

Principales características:

- Reutiliza tecnologías Java ya probadas como Hibernate y Spring bajo una interfaz simple y consistente.
- Patrones de visualización, potentes y fáciles de usar con GSP (Groovy Server Pages).
- Bibliotecas de etiquetas dinámicas para crear fácilmente componentes web.
- Diseñado según el paradigma Modelo Vista Controlador.
- Proporciona un entorno completo de desarrollo, incluyendo un servidor web y recarga automática de recursos.
- Entorno de desarrollo preparado para funcionar desde el primer momento.
- Funcionalidad disponible mediante métodos dinámicos.
- Desarrollo de aplicaciones extensibles mediante plugins.

2.1.1.2 Groovy

“Groovy es un lenguaje de programación orientado a objetos implementado sobre la plataforma Java. Tiene características similares a Python, Ruby, Perl y Smalltalk. Groovy usa una sintaxis muy parecida a Java, comparte el mismo modelo de objetos, de hilos y de seguridad. Desde Groovy se puede acceder directamente a todas las API existentes en Java. El bytecode generado en el proceso de compilación es totalmente compatible con el generado por el lenguaje Java para la Java Virtual Machine (JVM), por tanto puede usarse directamente

en cualquier aplicación Java. Groovy puede usarse también de manera dinámica como un lenguaje de scripting.” (14)
Entre los atractivos que distinguen a este lenguaje se pueden destacar los siguientes:

- Funciones modernas de programación disponibles para los desarrolladores de Java, con curva de aprendizaje casi nula
- Aumento de la productividad de los desarrolladores de código mediante la reducción y simplificación de la sintaxis si se compara con el mismo código Java necesario para ejecutar las mismas acciones.

2.1.1.3 Plugins y librerías

Los plugins y librerías son funcionalidades que se integran a la aplicación y facilitan el trabajo del programador al no tener que reinventar la rueda, son muy útiles debido a que son genéricos y facilitan tareas como la gestión de usuarios y permisos, la publicación de documentos en diferentes formatos y la generación de informes. Estas funcionalidades se incluyen en módulos que pueden ser cargados por el framework e integradas al sistema.

Mail Plugin

“El mail plug-in proporciona la capacidad de enviar e-mail a una aplicación Grails configurando un Spring MailSender sobre la base de los parámetros por defecto.” (15)
Mediante este plugin, configurándolo debidamente se podrán enviar notificaciones por correo electrónico desde la aplicación.

Spring Security Core Plugin

El plugin Spring Security simplifica la integración de Spring Security en las aplicaciones Grails. El plugin proporciona parámetros por defecto, con muchas opciones de configuración para su personalización. “Spring Security es un potente y altamente configurable framework de autenticación y control de acceso. Es el estándar para asegurar aplicaciones basadas en Spring. Fundado en 2003 y mantenido activamente por SpringSource, ya que, hoy en día se utiliza para proteger numerosos entornos muy exigentes, incluyendo organismos gubernamentales, aplicaciones militares y bancos centrales.”(16)

JFreeChart

“JFreeChart es una biblioteca de Java, 100% libre, que hace que sea fácil para los desarrolladores mostrar gráficos de calidad profesional en sus aplicaciones. Cuenta con un

amplio conjunto de características, JFreeChart incluye: una API consistente y bien documentada, el apoyo a una amplia gama de tipos de gráficos; un diseño flexible que es fácil de extender, y se dirige tanto del lado del servidor y las aplicaciones del lado del cliente; soporte para muchos tipos de salida, incluyendo los componentes Swing, archivos de imagen (incluyendo PNG y JPEG) y gráficos vectoriales formatos de archivo (incluyendo PDF, EPS y SVG)" (14) (17)

2.1.2 Entorno Integrado de Desarrollo

Un IDE (**i**ntegrated **d**evelopment **e**nvironment según sus siglas en inglés) es un programa que comprende un entorno de programación amigable para uno o varios lenguajes, brindando facilidades al programador, tales como: un editor de código, un compilador, un depurador y, opcionalmente, un constructor de interfaz gráfica.

2.1.2.1 IntelliJ IDEA

"Es un IDE centrado en el código y enfocado en la productividad del desarrollador. El editor entiende profundamente el código y conoce su camino alrededor del código base.

Completamiento de código rápido e inteligente

IntelliJ IDEA ofrece un inteligente y consciente completamiento de código. Sabe cuándo puede ser necesario convertir a un tipo, y también es consciente de las comprobaciones de tipo en tiempo de ejecución que ha realizado, después de lo cual puede llevar a cabo la conversión y la invocación de métodos en una sola acción.

Facilitar la edición multilingüe

IntelliJ IDEA reconoce fácilmente un idioma dentro de otro, las inyecciones de lenguaje, tales como SQL dentro de una aplicación Java, HTML en JavaScript, etc. Para las inyecciones el editor proporciona una apropiada asistencia de codificación en línea y un editor de fragmento de la inyección dedicado.

Nueva interfaz de usuario compatible con varias plataformas

La interfaz de usuario completamente rediseñada de IntelliJ IDEA es más ligera, flexible y personalizable. Toma ventajas del soporte del teclado totalmente, para una codificación diaria productiva." (15)(18)

El IntelliJ IDEA tiene una versión de código abierto, diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas con Windows y Linux. Este IDE soporta Grails como Framework, es una herramienta para programadores con completamiento de

código, pensada para escribir, compilar, depurar y ejecutar programas en una gran variedad de lenguajes de programación.

2.1.3 Servidor WEB

Un **servidor web** o **servidor HTTP** es un programa informático que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse generalmente el protocolo HTTP (Hypertext Transfer Protocol).

2.1.3.1 Apache Tomcat

Apache Tomcat es un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

“Apache Tomcat es una implementación de software de código abierto de la tecnología Java Servlet y JavaServer Pages. El Java Servlet y JavaServer Pages se han desarrollado bajo la Java Community Process.”(16)(19)

A partir de la versión 7.x, incorpora nuevas características: implementado de Servlet 3.0 y JSP 2.2, mejoras para detectar y prevenir "fugas de memoria" en las aplicaciones web, limpieza interna de código, soporte para la inclusión de contenidos externos directamente en una aplicación web.

Para el desarrollo del sistema se utilizará la ultima versión estable del Tomcat, la 7.0.27.

2.1.4 Sistema gestor de base de datos

2.1.4.1 PostgreSQL

Es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada y libre, o apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group por sus siglas en inglés).

"PostgreSQL es un potente sistema objeto-relacional de bases de datos de código abierto. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación por su fiabilidad, integridad de datos y la corrección. Se ejecuta en todos los principales sistemas operativos, incluyendo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows". (20)

Principales características:

- Alta concurrencia
- Amplia variedad de tipos nativos
- Funciones
- Escalabilidad

Para la solución se utilizará en su versión 9.1

2.2 SEPI

El ***S***istema para ***E***valuar ***P***royectos, la ***I*** por ser aplicado a proyectos Informáticos, ***SEPI,*** será una herramienta de soporte a la puesta en práctica del método MEPROI para evaluar proyectos y establecer un orden de prioridad que ayude a la toma de decisiones garantizando la confiabilidad del procesamiento estadístico en el que se sustenta.

2.2.1 Personas relacionadas

Las personas relacionadas con el sistema son aquellas que:

- Obtienen un resultado del valor de uno o varios procesos que se ejecutan en el mismo.
- Se encuentran involucradas en dichos procesos, pues participan en ellos, pero no obtienen ningún resultado de valor.

Basados en los actores que plantea el método y la modelación realizada para el sistema se identifican las personas relacionadas siguientes:

Tabla 3. Personas relacionadas con el sistema

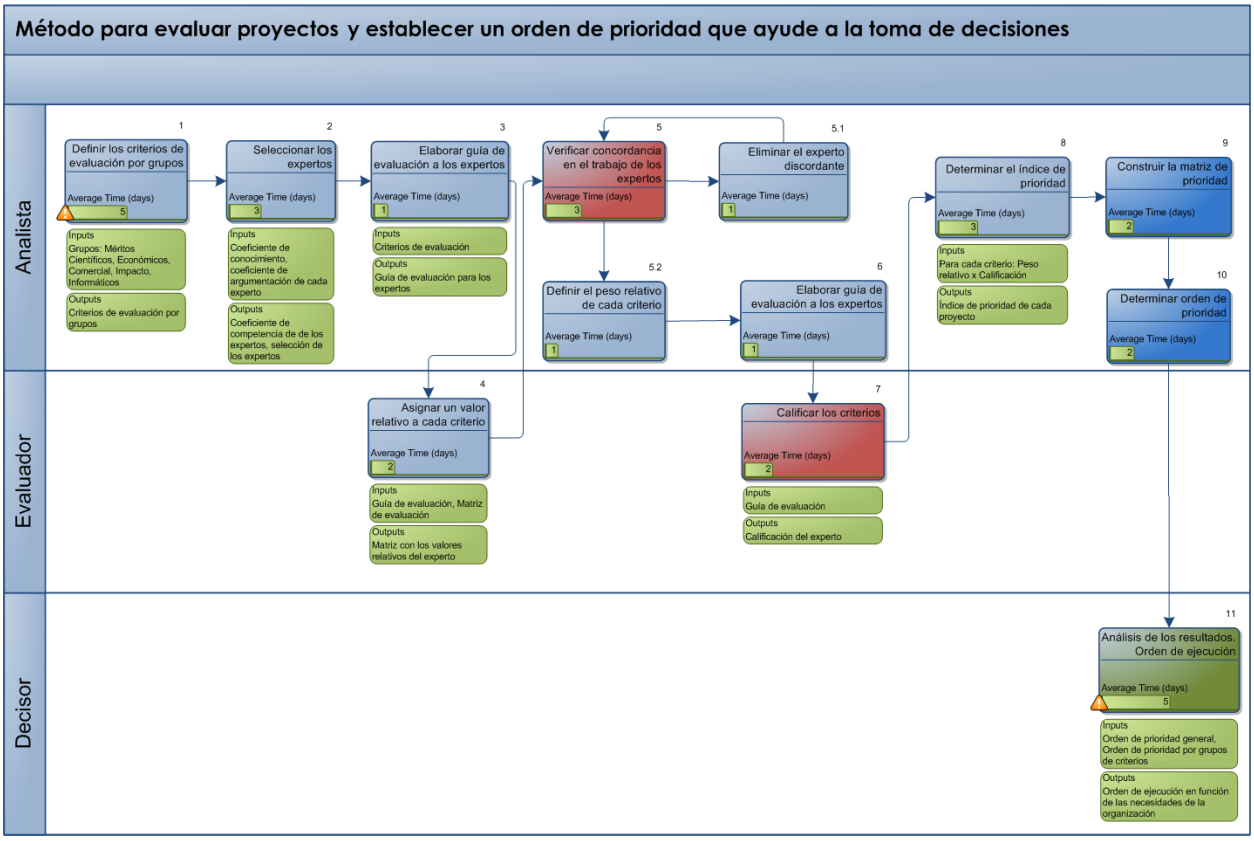
Personas relacionadas	Descripción
Analista	Dirigirá todo el proceso de evaluación, definirá los criterios utilizados, seleccionará los evaluadores y aplicará el método de evaluación para establecer el orden de prioridad de los proyectos.

Evaluador	Expresará la calidad del proyecto de acuerdo con sus conocimientos sobre la temática que trata, definirá el peso relativo de cada criterio.
Calificador	Calificará los criterios empleados según la escala pre-establecida en cada proyecto.
Decisor	Analizará los resultados obtenidos de la aplicación del método y aprobará el orden de ejecución de los proyectos de acuerdo a los intereses de la organización.

2.2.2 Flujo de actividades

A continuación se muestra el flujo de actividades del método MEPROI de acuerdo a cada uno de los actores que ejecutan estas actividades.

Figura 7. Flujo de actividades de MEPROI

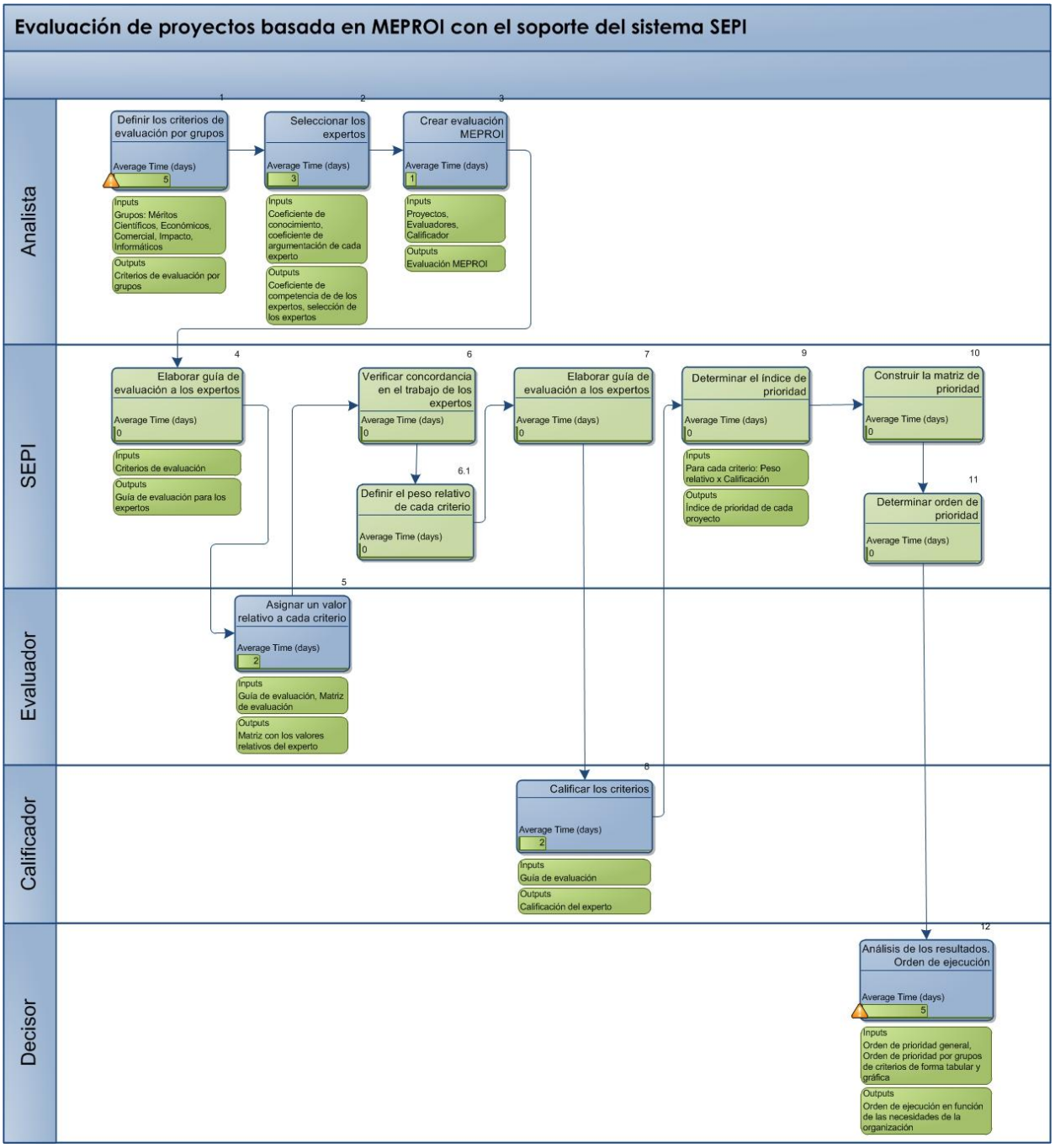


El analista como se puede observar lleva el trabajo de todo el procesamiento estadístico y la responsabilidad de los resultados arrojados.

Es importante destacar que la actividad *Eliminar el experto discordante* en caso de no existir concordancia no se aborda en esta versión inicial del sistema.

Con la utilización del sistema el flujo de actividades se simplificaría para cada persona relacionada:

Figura 8. Flujo de actividades de MEPROI utilizando SEPI



Con la utilización del sistema se garantiza la confiabilidad de las actividades que incluyen el procesamiento estadístico bajo el que se sustenta el método:

- Verificar concordancia en el trabajo de los expertos
- Determinar el índice de prioridad general y por grupos de criterios
- Determinar el orden de prioridad por grupo de criterios

2.2.3 Funcionalidades

Las funcionalidades del sistema abarcarán el flujo básico de la evaluación realizada por el método MEPROI y otros aspectos de interés.

A continuación se detallan:

- La autenticación se realizará contra la base de datos del sistema, lo cual asegurará que ninguna persona ajena a la realización de la evaluación tenga acceso a la aplicación.
- El analista compartirá el rol de administrador del sistema.
- Se mostrarán las funcionalidades y las vistas de acuerdo al rol del usuario que se encuentre autenticado en la aplicación.
- Permitirá gestionar toda la información referente a los usuarios, proyectos, criterios, grupos de criterios y evaluaciones que interactúan con la aplicación.
- Permitirá realizar la evaluación del proyecto por cada uno de los evaluadores.
- Será capaz de verificar la concordancia en el trabajo de los expertos.
- Permitirá realizar la calificación de los criterios en el proyecto por la persona determinada para ello.
- Obtendrá el índice de prioridad general y por grupos de criterios.
- Será capaz de mostrar los resultados obtenidos luego de la aplicación del método MEPROI en forma tabular y gráfica, como forma de ayuda a la comprensión de la información mostrada al decisor.

- Se mostrará al decisor los índices de prioridad de los proyectos involucrados en la evaluación, ordenados por índice de prioridad y los índices de prioridad por grupos de criterios para cada uno de los proyectos.
- Será capaz de enviar correos electrónicos, usando para ello el SMTP de la UCI, así como un usuario y contraseña autorizados para autenticarse. El envío de estos correos desde la aplicación será muy útil, ya que permitirá al analista notificar a los evaluadores y a los decisores de alguna nueva evaluación o de otras cuestiones de su interés.

2.3 Historias de usuarios

Las historias de usuarios (HU) son la técnica utilizada en XP para describir los requisitos del software con pequeños textos en los que el cliente describe una actividad que realizará el sistema de forma sencilla y clara. Se puede considerar que las historias de usuario juegan un papel similar a lo que equivaldría a los casos de uso en el proceso unificado, pero son muy diferentes, pues muestran solamente la silueta de una tarea a realizarse.

Leyenda:

Número: Número de id para las HU, sería incremental en el tiempo.

Nombre Historia de Usuario: Es el nombre de la HU, sirve para identificarla fácilmente tanto para los desarrolladores coma para los clientes.

Modificación de Historia de Usuario Número: Cantidad de modificaciones que se le ha realizado a la historia de usuario (de no tener modificaciones se pone ninguna, si no la cantidad de veces que ha sido modificada).

Usuario: Nombre del programador encargado de implementar la HU.

Prioridad del negocio: Qué tan importante es para el cliente, se clasifica en Muy alta, Alta y Media.

Riesgo de desarrollo: Qué tan difícil es para el desarrollador (Alto, Medio o Bajo).

Iteración asignada: Iteración a que corresponde, número de la iteración en la que se desarrollará la HU.

Puntos estimados: Tiempo en semanas que se le asignará. (Estimado)

Descripción: Es la descripción de la historia, detallando las operaciones del usuario y las respuestas del sistema.

Observaciones: Informaciones de interés, como glosarios, detalles del usuario, etc.

Prototipo de Interfaz: Contiene la imagen de una de las interfaces de usuario relacionadas con la HU.

A continuación se muestra la historia de usuario Autenticar usuario, las restantes se encuentran en el ***Anexo 1***.

Historia de Usuario	
Número: HU-1	**Nombre Historia de Usuario:** Autenticar usuario.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada:** Primera.
Prioridad en el negocio: Muy alta.	**Puntos Estimados: 1**
Descripción: La persona interesada se autenticará en el sistema introduciendo un usuario y una contraseña. El sistema verifica que todos los datos necesarios hayan sido insertados de forma correcta, dando acceso al sistema según el rol y de ello dependerá la sesión correspondiente.	

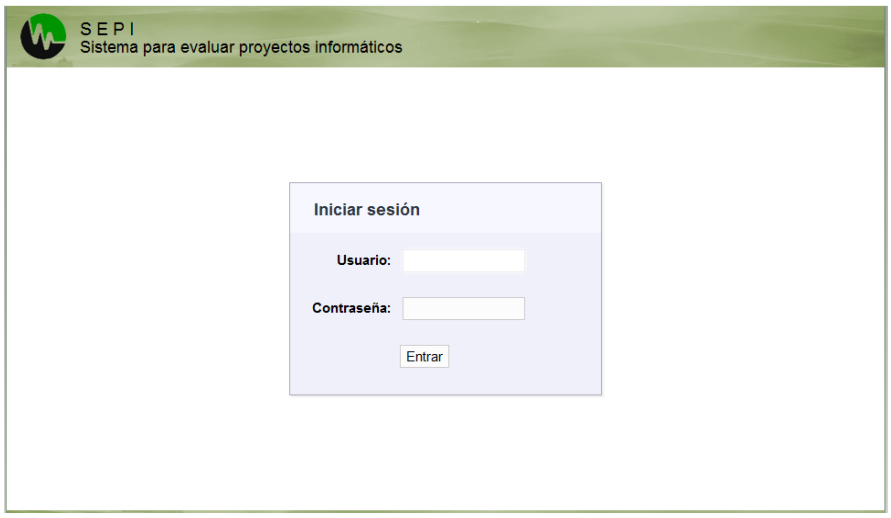
Observaciones:
Prototipo de Interfaz: 

Tabla 4. HU-1: Autenticar usuario

2.4 Requisitos no funcionales

Los requisitos no funcionales son una característica requerida del sistema, del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto del desarrollo, que señala una restricción del mismo.

Software

- Se utilizará cualquier navegador para acceder al sistema, se recomienda Mozilla Firefox.
- Se utilizará un servidor de base de datos con SGBD PostgreSQL 9.1.
- Se utilizará un servidor de aplicaciones Apache Tomcat 7.0.27.
- El sistema podrá ser utilizado en los sistemas operativos Windows y Linux.

Hardware

- El servidor de base de datos deberá tener como mínimo 512 MB de RAM.
- El servidor de aplicaciones deberá tener como mínimo 512 MB de RAM.

Apariencia o interfaz externa

- El sistema debe contar con una interfaz sencilla, fácil y cómoda que permita a los usuarios interactuar con el sistema, aún sin tener altos conocimientos del funcionamiento de este.
- Las interfaces evitarán cargar diálogos, paneles o ventanas innecesarias.
- La fuente de la letra deberá ser uniforme para todas las interfaces.
- Debe existir coherencia entre los íconos, los botones y la función que deben realizar.

Seguridad

- El sistema contará con niveles de acceso hacia los usuarios, los cuales se realizarán acorde con los roles autorizados para el uso de cada funcionalidad tanto a nivel de funciones de las aplicaciones como de información de la base de datos.

2.5 Estimación de esfuerzo por Historia de Usuario

La tabla que se muestra a continuación contiene la estimación de esfuerzo por Historia de Usuario según el orden a realizar. Esta estimación incluye todo el esfuerzo asociado a la implementación de la HU, la misma expresa utilizando como medida el punto (máximo esfuerzo). Un punto se considera como una semana ideal de trabajo, donde se trabaje el tiempo planeado sin ningún tipo de interrupción. En el caso especificado seguidamente se decidió por parte del equipo de desarrollo dedicar todo el esfuerzo posible por el número de HU identificadas.

Historia de Usuario	Puntos estimados
Autenticar usuario	1
Gestionar Usuario	1
Gestionar Proyecto	1
Gestionar Criterio	1
Gestionar Grupo de Criterio	1
Gestionar Evaluación Meproi	1
Realizar Evaluación Peso-Criterios	1
Verificar Concordancia Expertos	1
Realizar Calificación Criterios	1
Calcular Índice de Prioridad	1
Graficar Resultados Evaluación	1

Enviar Notificaciones por Correo Electrónico	1

Tabla 5. Estimación de esfuerzo por Historia de Usuario

2.6 Plan de iteraciones

Luego de identificar las HU y la estimación del esfuerzo por cada una de ellas, se procede a realizar el plan de iteraciones, en el cual estarán contenidas las HU en el orden a realizar por cada iteración según su orden de relevancia, así como el total de semanas que durarán cada una de estas. Teniendo en cuenta el riesgo para desarrollar cada una de las historias de usuario, el tamaño del equipo de desarrollo, así como otros factores subjetivos, se decidió dividir el proyecto en tres iteraciones, detalladas a continuación:

Iteración 1: En esta iteración se implementarán las historias de usuarios de prioridad muy alta, funcionalidades que son indispensables para cubrir las necesidades del cliente y que inciden críticamente en el funcionamiento de la aplicación.

Iteración 2: En esta iteración se implementarán las historias de usuarios de prioridad alta, las cuales no son menos importantes que las mencionadas anteriormente pero debido al número de HU se determinó realizarlas en otra iteración.

Iteración 3: En esta iteración se implementarán las historias de usuarios de prioridad media.

Iteraciones	Orden de las HU por iteración	Duración total de la iteración en semanas
Iteración 1	Autenticar usuario	20
	Gestionar usuario	
	Gestionar proyecto	
	Gestionar criterio	
	Gestionar grupo de criterio	
	Gestionar evaluación Meproi	
Iteración 2	Realizar evaluación peso-criterios	13
	Realizar calificación criterios	
	Verificar concordancia expertos	
	Calcular índice de prioridad	
Iteración 3	Graficar resultados evaluación	5
	Enviar notificaciones vía correo electrónico	

Tabla 6. Plan de Iteraciones

2.7 Plan de entregas

Este plan se obtiene a través de reuniones entre el equipo de trabajo y los clientes, para definir el marco temporal de la realización del sistema. Según la duración de cada iteración, así será la fecha aproximada de entrega. En el caso de la solución propuesta, se acordó y determinó entre el cliente y el equipo de desarrollo, que debido al número de HU a implementar las fechas de entrega serán reducidas, sin dejar de tener presente el grado de dificultad de las HU.

A continuación se muestra una tabla con las fechas aproximadas de entregas por cada iteración.

Herramienta	**Final 1ra Iteración** **4ta semana de marzo**	**Final 2da Iteración** **3ra semana de abril**	**Final 3ra Iteración** **2da semana de mayo**
SEPI	0.1	0.2	1.0

Tabla 7. Plan de entregas

2.8 Conclusiones parciales

- Se establecieron las tecnologías y herramientas para el desarrollo: Grails como marco de trabajo en su versión 2.0.3, PostgreSQL 9.1 como sistema gestor de base de datos, Apache Tomcat 7.0.27 como servidor de aplicaciones y el IntelliJ IDEA 11.0 como IDE de desarrollo.
- Se describieron las fases de Exploración y Planeación para la propuesta de solución obteniendo los artefactos necesarios para la construcción: HU, plan de iteración, el esfuerzo estimado por HU y el plan de entregas.

Capítulo 3: Iteraciones y Producción

En el presente capítulo se procederá siguiendo la filosofía planteada por la metodología XP, según la cual la implementación debe realizarse de forma iterativa e incremental, obteniendo al final de cada iteración un producto funcional que debe ser examinado de conjunto con el cliente, pues de esta manera se garantiza la retroalimentación entre los desarrolladores y el cliente. En el presente capítulo se puntualizan las iteraciones llevadas a cabo durante la etapa de construcción del sistema, definiéndose las tarjetas CRC (Clase, Responsabilidad, Colaborador) como herramienta muy útil para propiciar el enfoque orientado a objetos y exponiéndose las tareas generadas por cada historia de usuario. Por último se realizarán las pruebas que contribuyen a la calidad de la aplicación en general.

3.1 Descripción de los algoritmos utilizados

3.1.1 Verificar concordancia en el trabajo de los expertos

La concordancia en los criterios de los expertos es una funcionalidad de relevancia en el sistema. A través de ella se puede conocer si existe consenso entre los expertos que realizan la evaluación.

"Es importante verificar el consenso de los expertos para tener confianza en los datos y continuar con el proceso de evaluación.

Se utiliza el coeficiente de concordancia de Kendall (W) para verificar la consistencia en el trabajo de los expertos con el uso de ligas pues un mismo experto proporciona el mismo peso para diferentes criterios, con la introducción de este elemento la expresión es la siguiente:

$$W = \frac{12 \sum_{j=1}^{C} (S_j - \bar{S})^2}{E^2(C^3 - C) - E \sum_{i=1}^{E} T_i}$$

donde:

C: Número de criterios a evaluar

E: Número de expertos involucrados

S_i: Refleja la suma de rangos correspondientes a la evaluación realizada por los expertos a la pregunta j y se define según la expresión:

$$S_j = \sum_{i=1}^{E} V_{ij}$$

V_{ij}: Es el rango asociado a la evaluación del experto "i" a la pregunta "j"

$\bar{S}$: media de la suma de rangos de cada pregunta j

$$\bar{S} = \frac{\sum_{j=1}^{C} S_j}{C}$$

T_i: Resultado de los rangos iguales, llamados también ligas, que ofreció el experto i para las preguntas y se define como sigue:

$$T_i = \frac{\sum_{i=1}^{l} (t^3 - t)}{12}$$

l: Número de grupos con rangos iguales para el experto i.

t: Número de observaciones dentro de cada uno de los grupos para el experto i.

Teniendo W (coeficiente de concordancia de Kendall) se procede a calcular el Chi cuadrado real:

$$\mathrm{X}^2 = \mathrm{E}(\mathrm{C} - 1)\mathrm{W}$$

El Chi cuadrado calculado se compara con el tabulado en la tabla del percentil de la distribución Chi-cuadrado:

Si se cumple:

$X^2_{real} > X^2_{(\propto, C-1)}$ $\therefore$ se infiere que existe concordancia de criterios entre los expertos al considerar válida la hipótesis alternativa H1: existe concordancia entre los expertos.

Para tener un 95% de confianza se utilizará $\alpha = 0.05$.” (3)

3.1.2 Determinar el índice de prioridad

Con la definición del peso relativo para cada uno de los criterios a partir del consenso en el trabajo de los expertos y la calificación realizada en una escala de 1-5 del comportamiento de cada criterio en el proyecto se construye la matriz de calificación a través de la cual se determina el índice de prioridad.

Figura 9. Matriz para la calificación de los criterios en el proyecto

Criterios	Calificación (c)	P	$P\ x\ c$
C_1	c_1	Ep_1	$Ep_{1x}\, c_1$
C_2	c_2	Ep_2	$Ep_{2x}\, c_2$
C_3	c_3	Ep_3	$Ep_{3x}\, c_3$
C_n	c_n	Ep_n	$Ep_{nx}\, c_n$
			$\sum P\ x\ c$

El índice de prioridad puede definirse de la siguiente manera:

IP = Índice de prioridad del proyecto

$$IP = \frac{\sum P\ x\ c}{5}$$

5: valor máximo para la calificación.

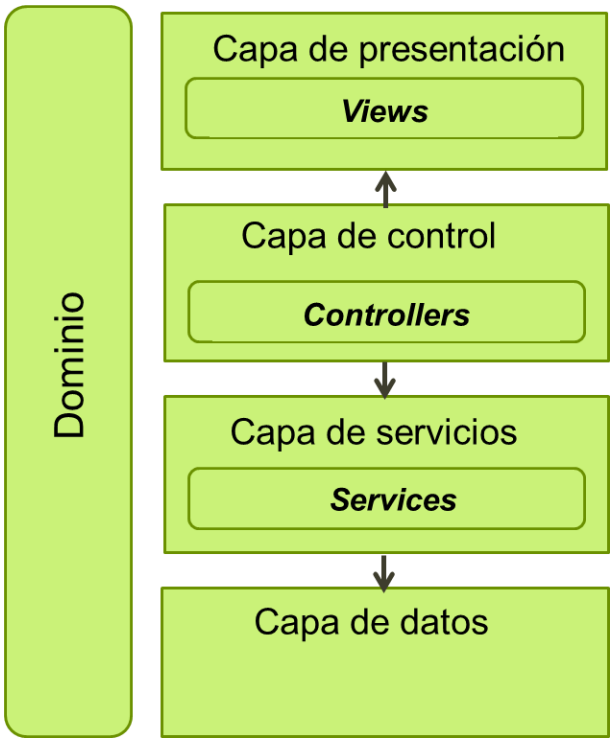
3.2 Arquitectura del sistema

La arquitectura de software es la organización fundamental de un sistema encarnado en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (21). Toda arquitectura de software debe describir diversos aspectos del software, y generalmente, cada uno de estos se describe de una manera más comprensible si se utilizan distintos modelos o vistas.

Para el sistema se adoptó una arquitectura en capas, dichas capas son: la Capa de Presentación, que contiene las vistas de la aplicación y es la encargada de la comunicación con el usuario; la capa de Control, la cual alberga los controladores existentes y es la encargada de proporcionar la comunicación entre las vistas y la lógica de negocio, presente en los servicios; la capa de Servicios, que contiene las entidades encargadas de manejar la lógica de negocio de la aplicación y la capa de Datos, cuya función es contener las entidades

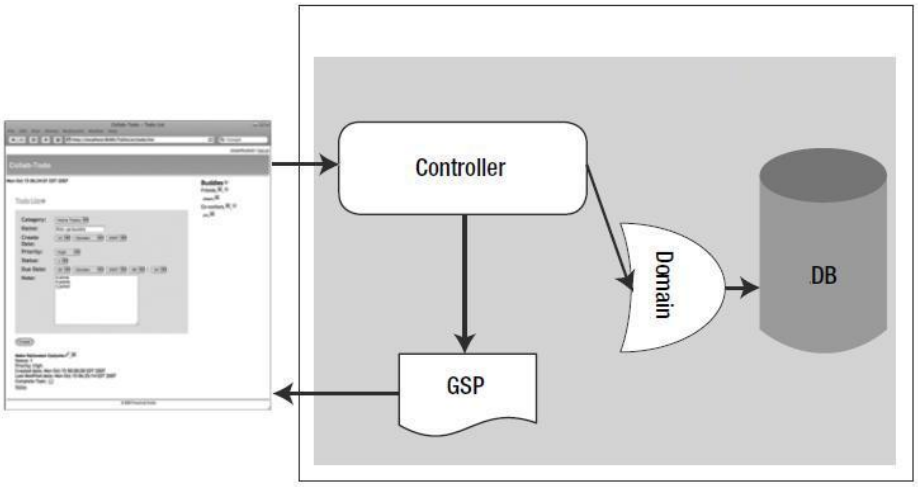
persistentes. Adicionalmente se define una capa de Dominio que es transversal al resto de las capas, ya que contiene los objetos que sirven de comunicación entre las capas. Se diferencia la capa de Datos de la capa de Dominio, pues no todos los elementos del dominio son entidades persistentes.

Figura 10. SEPI: Arquitectura en capas



El patrón Modelo-Vista-Controlador (Model-View-Controller) es un estilo arquitectónico empleado en la construcción de aplicaciones web. El mismo separa la parte lógica de una aplicación de su presentación. Grails implementa el patrón MVC, en el que la lógica del negocio se separa de la presentación de la aplicación. Esto le permite cambiar fácilmente el aspecto de la aplicación sin accidentalmente modificar su comportamiento.

Figura 11. Arquitectura de una aplicación Grails



Modelo: Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

Específicamente, el framework utilizado (Grails), se vale de las bondades del patrón **MVC** brindando las vistas en forma o extensión gsp. Este tipo de páginas conforman la capa de interacción con el cliente. Los controladores, encargados de mantener el flujo de comunicación entre las vistas y el modelo, están definidos por los groovy controllers o controladores groovy, los cuales especifican cómo acceder a los datos y el envío de las respuestas a las vistas. El modelo, definido por clases de dominio de groovy que se mapean en la base de datos utilizada, complementan la tercera capa y permiten el manejo y almacenamiento de los datos. De esta forma, queda definida la utilización del patrón **MVC** en la interacción y desarrollo con el framework.

3.2.1 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

- **Inversión del control (IoC):** Es un principio de programación donde el aspecto que se invierte es el modo en que los objetos obtienen las instancias de los objetos que colaboran con él o de los cuales depende. (20)(22)

La inversión de control es un patrón utilizado en Grails, según el cual las dependencias de un componente no deben gestionarse desde el propio componente para que éste sólo contenga la lógica necesaria para hacer su trabajo.

Al utilizar servicios en Grails, solo se escribe una variable con su nombre, sin tener en cuenta la instanciación del servicio y su configuración. Grails configura Spring para que gestione su ciclo de vida y sus dependencias. El objetivo de esta técnica es mantener los componentes lo más sencillos que sea posible, incluyendo únicamente código que tenga relación con la lógica de negocio. Así, la aplicación será más fácil de comprender y mantener.

3.3.1.1 Patrones GRASP

“Son patrones generales de software para asignación de responsabilidades, es el acrónimo de "General Responsibility Assignment Software Patterns". Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software “. (18)(23)

- **Patrón Controlador**: Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

- **Alta cohesión:** Se refiere a que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la ella misma.

- **Bajo acoplamiento:** Se refiera a 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.

- **Experto**: 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. De esta manera se garantiza una alta cohesión y un bajo acoplamiento.

3.3.1.2 Patrones GOF (Gang of Four)

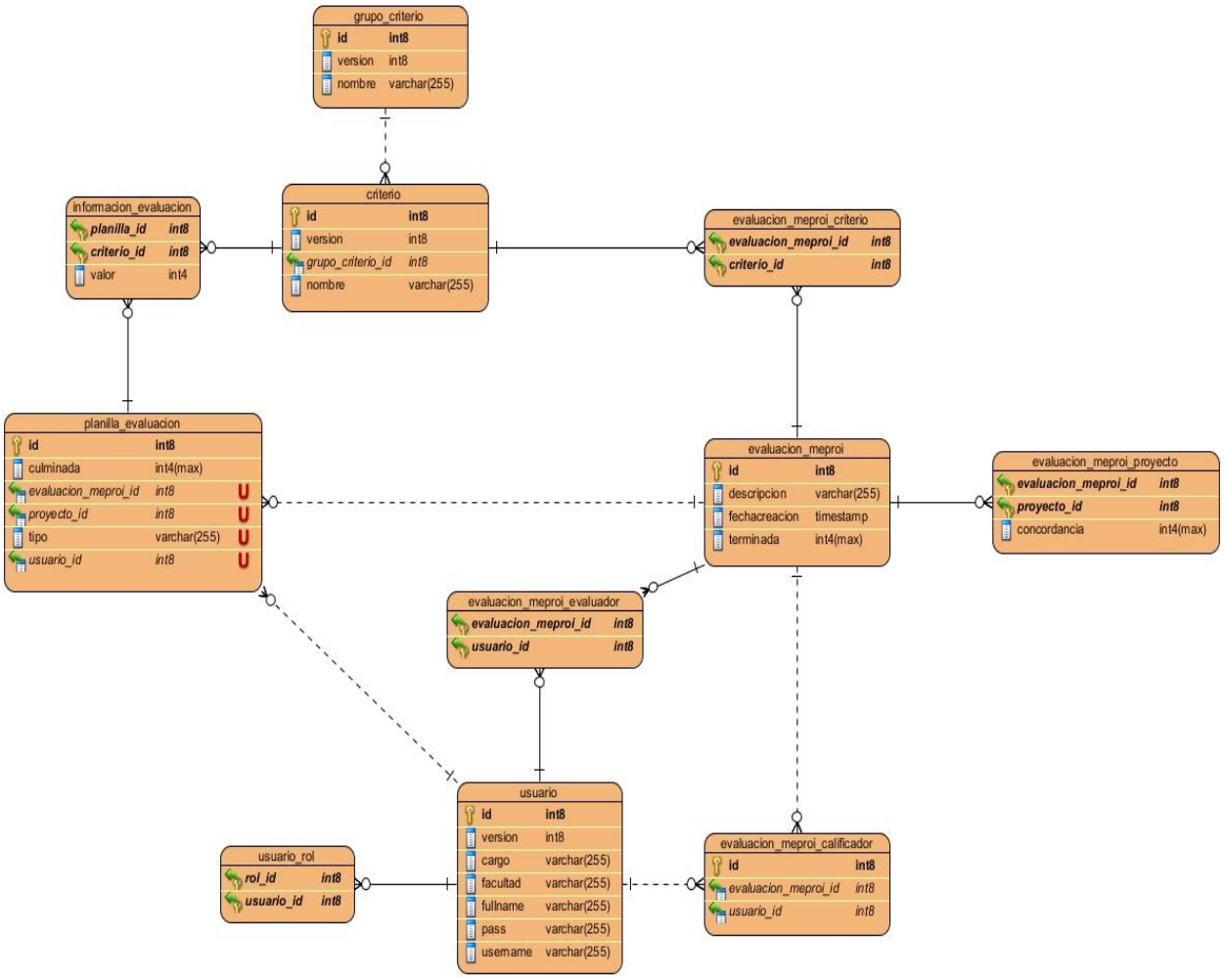
- **Singleton:** Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. Se utiliza en la instanciación de los servicios en los controladores. Por defecto en Grails todos los servicios son Singleton.

- **Decorador / Decorator:** Añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. El uso de este patrón se centra en la utilización del SiteMesh integrado a la arquitectura del framework. El mismo permite partir de una plantilla inicial de la página incorporar modificaciones a partir de cada vista particular.

3.3 Modelo de datos

El modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Este describe las estructuras de datos de la base correspondiente al contenido del sistema, permite describir los elementos que intervienen y la forma en que se relacionan estos elementos entre sí.

Figura 12. Modelo de datos SEPI



3.4 Tarjetas CRC (Clase, Responsabilidad, Colaborador)

La principal tarea de las tarjetas CRC es propiciar el enfoque orientado a objetos y dejar el pensamiento procedimental. Las tarjetas están divididas en tres secciones: nombre de la clase, responsabilidades y colaboradores.

Una clase describe cualquier objeto o evento, mediante los atributos y los métodos, las responsabilidades son la las tareas que realizan o los métodos correspondientes a la clase y los colaboradores son las demás clase con las que trabaja conjuntamente para cumplir con sus responsabilidades.

UsuarioController

Responsabilidad	Colaborador
Listar Crear Mostrar Modificar Eliminar	Usuario

Tabla 8. Tarjeta CRC Usuario

ProyectoController	
Responsabilidad	**Colaborador**
Listar Crear Mostrar Modificar Eliminar	Proyecto

Tabla 9. Tarjeta CRC Proyecto

GrupoCriterioController	
Responsabilidad	**Colaborador**
Listar Crear Mostrar Modificar Eliminar	GrupoCriterio

Tabla 10. Tarjeta CRC Grupo Criterio

CriterioController	
Responsabilidad	**Colaborador**
Listar Crear Mostrar Modificar Eliminar	Criterio

Tabla 11. Tarjeta CRC Criterio

EvaluacionMeproiController	
Responsabilidad	**Colaborador**

Listar Crear Mostrar MostrarPlanilla Modificar Eliminar	EvaluacionMeproi EvaluacionMeproiService

Tabla 12. Tarjeta CRC Evaluación Meproi

CorreoController	
Responsabilidad	**Colaborador**
Enviar Correo	Mail Plugin

Tabla 13. Tarjeta CRC Correo

EvaluadorController	
Responsabilidad	**Colaborador**
Meproi Proyectos Evaluacion Evaluar Mostrar	EvaluadorService ConcordanciaService

Tabla 145. Tarjeta CRC Evaluador

CalificadorController	
Responsabilidad	**Colaborador**
Meproi Proyectos Calificacion Calificar Mostrar	CalificadorService ConcordanciaService

Tabla 15. Tarjeta CRC Calificador

ResultadoController	
Responsabilidad	**Colaborador**
Listar Resultados Pdf	PrioridadService GraficoService PdfRenderingService

Tabla 167. Tarjeta CRC Resultado

LoginController	
Responsabilidad	**Colaborador**
Auth AuthAjax Denied AuthFail	SpringSecurityService

Tabla 178. Tarjeta CRC Login

3.5 Tareas de ingeniería

Para definir las tareas de ingeniería se cuenta con una plantilla, la cual permite definir cada una de las actividades que estarán asociadas a las historias de usuario y que permitirán su implementación. A continuación se muestran las tareas asignadas por iteraciones.

3.5.1 Iteración 1

Durante esta iteración se abordaron las Historias de Usuarios de máxima prioridad donde se construyó la base de la arquitectura del producto con las funcionalidades primarias necesarias para ser mostrado al cliente y obtener una rápida y amplia retroalimentación. A continuación se describen en la siguiente tabla:

Historias de Usuario	Estimación	Real
Autenticar usuario	1	1
Gestionar usuario	1	1
Gestionar proyecto	1	1
Gestionar criterio	1	1
Gestionar grupo criterio	1	1
Gestionar evaluación Meproi	1	1

Tabla 189. HU implementadas en la Iteración 1

Las tareas a desarrollar por cada una de las historias de usuario anteriores son:

Historias de Usuario	Tareas de Ingeniería
Autenticar usuario	Diseño de la interfaz de autenticación Validar datos

Gestionar usuario	Crear usuario Eliminar usuario Modificar usuario Mostrar usuario
Gestionar proyecto	Crear proyecto Eliminar proyecto Modificar proyecto Mostrar proyecto
Gestionar criterio	Crear criterio Eliminar criterio Modificar criterio Mostrar criterio
Gestionar grupo criterio	Crear grupo criterio Eliminar grupo criterio Modificar grupo criterio Mostrar grupo criterio
Gestionar evaluación Meproi	Crear evaluación Meproi Eliminar evaluación Meproi Modificar evaluación Meproi Mostrar evaluación Meproi

Tabla 19. Tareas de Ingeniería para la Iteración 1

A continuación se detallan las tareas de ingeniería relacionadas con la HU-Autenticar Usuario para la *Iteración 1*, el resto de las tareas de esta iteración se muestran en el ***Anexo 2***.

Tarea de Ingeniería	
Número de la tarea: 1	**Número de Historia de Usuario**: 1
Nombre de la tarea: Diseño de la interfaz de autenticación	
Tipo de tarea: Desarrollo	**Puntos Estimados**: 0.5
Fecha inicio: 3/3/2012.	**Fecha fin**: 5/3/2012
Programador responsable: Leandro José Rodríguez Hernández.	

Descripción: Se implementa el diseño de la interfaz de autenticación, con los elementos necesarios para su correcto funcionamiento.

Tabla 201. HU-1.Tarea de ingeniería No.1: Diseño de la Interfaz de autenticación

Tarea de Ingeniería	
Número de la tarea: 2	**Número de Historia de Usuario**: 1
Nombre de la tarea: Validar Datos	
Tipo de tarea: Desarrollo	**Puntos Estimados**: 0.5
Fecha inicio: 3/3/2012.	**Fecha fin**: 5/3/2012
Programador responsable: Leandro José Rodríguez Hernández.	
Descripción: Se implementan las funcionalidades que permiten la captura de los datos, se verifican que sean correctos con la base de datos de la aplicación. En caso que sea correcto se devuelve el usuario a la página correspondiente según su privilegio.	

Tabla 212. HU-1.Tarea de ingeniería No.2: Validar Datos

3.5.2 Iteración 2

Durante esta iteración se abordaron las Historias de Usuarios de mayor prioridad donde se definen las tareas de desarrollo de las mismas. Al culminar se consta de un producto casi listo para su puesta en funcionamiento con la mayoría de sus funcionalidades más críticas ya implementadas. A continuación se describe en la siguiente tabla:

Historias de Usuario	Estimación	Real
Realizar evaluación peso-criterios	1	1
Verificar concordancia expertos	1	1
Realizar calificación criterios	1	1
Calcular índice de prioridad	1	1

Tabla 223. HU implementadas en la Iteración 2

Las tareas a desarrollar por cada una de las historias de usuario anteriores son:

Historias de Usuario	Tareas de Ingeniería
Realizar evaluación peso-criterios	Diseño del formulario de evaluación Procesar datos evaluación
Verificar concordancia expertos	Verificar concordancia
Realizar calificación criterios	Diseño del formulario de calificación Procesar datos calificación

Calcular índice de prioridad	Calcular prioridad

Tabla 234. Tareas de Ingeniería para la Iteración 2

A continuación se detallan las tareas de ingeniería relacionadas con la HU-Realizar evaluación peso-criterios para la *Iteración 2*, el resto de las tareas de esta iteración se muestran en el ***Anexo 3.***

Tarea de Ingeniería	
Número de la tarea: 1	**Número de Historia de Usuario**: 7
Nombre de la tarea: Diseño del formulario de evaluación	
Tipo de tarea: Desarrollo	**Puntos Estimados**: 0.5
Fecha inicio: 4/4/2012.	**Fecha fin**: 6/4/2012
Programador responsable: Leandro José Rodríguez Hernández.	
Descripción: Se implementa el diseño del formulario de evaluación, con los elementos necesarios para su correcto funcionamiento.	

Tabla 245. HU-7.Tarea de ingeniería No.1: Diseño del formulario de evaluación

Tarea de Ingeniería	
Número de la tarea: 2	**Número de Historia de Usuario**: 7
Nombre de la tarea: Procesar datos evaluación	
Tipo de tarea: Desarrollo	**Puntos Estimados**: 0.5
Fecha inicio: 4/4/2012.	**Fecha fin**: 6/4/2012
Programador responsable: Leandro José Rodríguez Hernández.	
Descripción: Se implementan las funcionalidades que permiten la captura de los datos, se verifican que sean correctos. En caso que sean correctos se procede a almacenarlos en la base de datos.	

Tabla 256. HU-7.Tarea de ingeniería No.2: Procesar datos evaluación

3.5.3 Iteración 3

En el transcurso de esta iteración se implementan las Historias de Usuarios referente a la tercera iteración que involucra facilidades que brinda la aplicación. Al culminar esta, se consta de un producto listo para su puesta en funcionamiento.

Historias de Usuario	**Estimación**	**Real**
Graficar resultados evaluación	1	1
Enviar notificaciones vía correo electrónico	1	1

Tabla 267. HU implementadas en la Iteración 3

Las tareas a desarrollar por cada una de las historias de usuario anteriores son:

Historias de Usuario	Tareas de Ingeniería
Graficar resultados evaluación	Graficar resultados
Enviar notificaciones vía correo electrónico	Diseñar interfaz de notificación Enviar notificación

Tabla 278. Tareas de Ingeniería para la Iteración 3

A continuación se detallan las tareas de ingeniería relacionadas con las HU para la *Iteración 3*.

Tarea de Ingeniería	
Número de la tarea: 1	**Número de Historia de Usuario**: 11
Nombre de la tarea: Graficar resultados	
Tipo de tarea: Desarrollo	**Puntos Estimados**: 1
Fecha inicio: 25/4/2012.	**Fecha fin**: 27/4/2012
Programador responsable: Leandro José Rodríguez Hernández.	
Descripción: Se hace uso de la librería auxiliar para graficar y se implementan los artefactos necesarios para su uso.	

Tabla 289. H-11.Tarea de ingeniería No.1: Graficar resultados

Tarea de Ingeniería	
Número de la tarea: 1	**Número de Historia de Usuario**: 12
Nombre de la tarea: Diseñar interfaz de notificación	
Tipo de tarea: Desarrollo	**Puntos Estimados**: 0.5
Fecha inicio: 25/4/2012.	**Fecha fin**: 27/4/2012
Programador responsable: Leandro José Rodríguez Hernández.	
Descripción: Se implementa el diseño de la interfaz de notificación, con los elementos necesarios para su correcto funcionamiento.	

Tabla 29. H-12.Tarea de ingeniería No.1: Diseñar interfaz de notificación

Tarea de Ingeniería	
Número de la tarea: 2	**Número de Historia de Usuario**: 12
Nombre de la tarea: Enviar Notificación	
Tipo de tarea: Desarrollo	**Puntos Estimados**: 0.5

Fecha inicio: 25/4/2012.	**Fecha fin**: 27/4/2012
Programador responsable: Leandro José Rodríguez Hernández.	
Descripción: Se configura el plugin auxiliar para enviar correos electrónicos y se implementan los artefactos necesarios para su uso**.**	

Tabla 30. H-12.Tarea de ingeniería No.2: Enviar notificación

3.6 Pruebas al sistema

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones específicas. Los resultados son observados, registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. La prueba está enfocada principalmente en la evaluación y determinación de la calidad del producto. Estas están definidas en estrategias, niveles, tipos, métodos y técnicas de prueba.

3.6.1 Estrategia de prueba

"La realización de pruebas en XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores y pruebas de aceptación o pruebas funcionales, destinadas a evaluar si al final de una iteración se logró implementar las funcionalidades requeridas por el cliente final." (24)

La estrategia de prueba describe el enfoque y los objetivos generales de las actividades de prueba. Incluye los niveles de prueba (unidad, integración, etc.) a ser diseccionados y el tipo de prueba a ser ejecutadas (funcional, stress, etc.).

La estrategia de prueba define:

- Técnicas de pruebas (manual o automática) y herramientas a ser usadas.
- Qué criterios de éxitos y culminación de la prueba serán usados.
- Consideraciones especiales afectadas por requerimientos de recursos o que tengan implicaciones en la planificación.

La estrategia a seguir para la realización de las pruebas al sistema contempla dos niveles de pruebas. El nivel de pruebas de Unidad y el Nivel de pruebas de Aceptación.

Prueba de Unidad: Enfocada a los elementos testeables (probables) más pequeños del software. Aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca.

Prueba de Aceptación: Las pruebas de aceptación son pruebas de caja negra, que representan un resultado esperado de determinada transacción con el sistema. Para que una historia de usuario se considere aprobada, deberá pasar todas las pruebas de aceptación elaboradas para dicha historia.

Es importante resaltar la diferencia entre las pruebas de aceptación y las unitarias en lo que al papel del usuario se refiere. Mientras que en las pruebas de aceptación juega un papel muy importante seleccionando los casos de prueba para cada historia de usuario e identificando los resultados esperados, en las segundas no tiene ninguna intervención por ser de competencia del equipo de programadores.

3.6.1.1 Pruebas de unidad

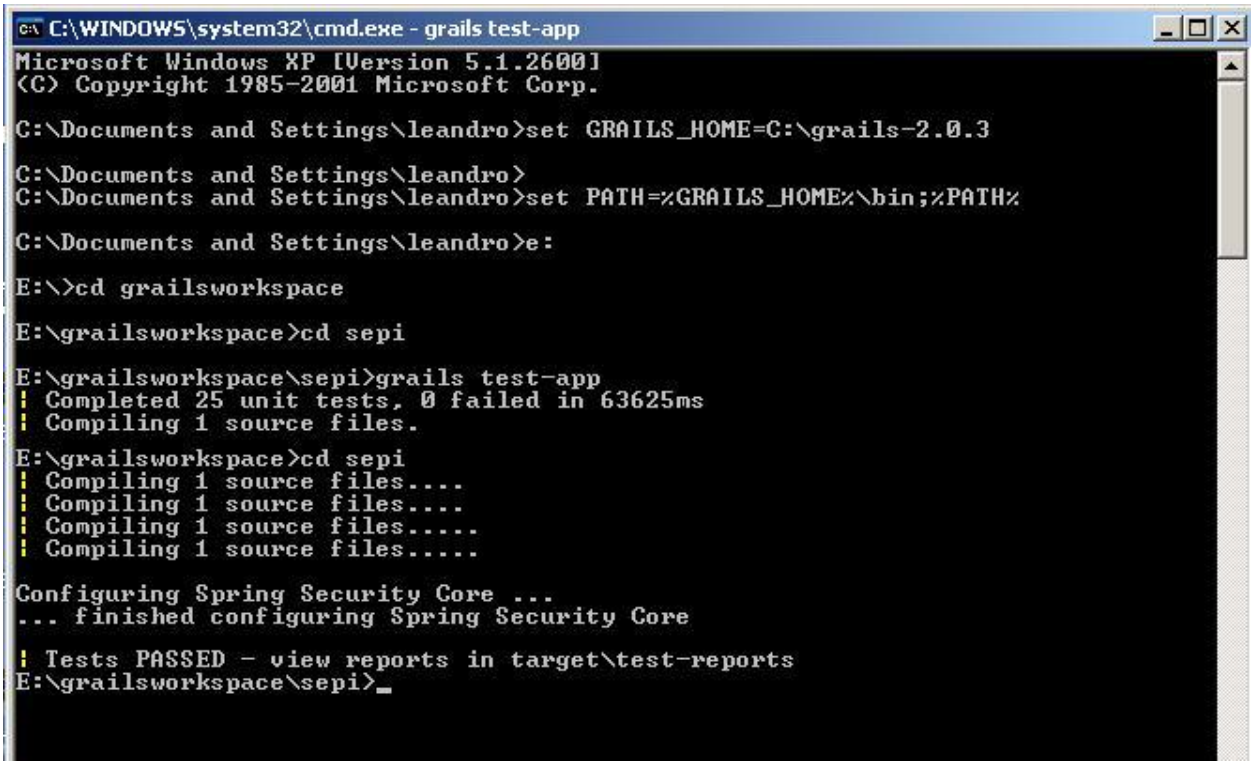
A su vez en el primer nivel, se realizaron pruebas automáticas haciendo uso del potente sistema de Testing o Prueba que brinda el propio framework utilizado. El segundo nivel de pruebas incluye la técnica manual incluyendo los tipos de prueba de funcionalidad y usabilidad basándose en los requerimientos tomados de las historias de usuario.

Los test unitarios son aquellos en los que se verifica que un método se comporta como debería, sin tener en cuenta su entorno. Esto significa que cuando se ejecutan las pruebas unitarias de un método Grails no inyectará ninguno de los métodos dinámicos con los se cuenta la aplicación cuando se está ejecutando. Así que, cuando se trabaja con entidades o servicios son los desarrolladores los responsables de crear y gestionar todos los objetos.

Durante el proceso de desarrollo, podemos probar el estado de la aplicación mediante el comando:

```
grails test-app
```

Figura 13. Resultados de las pruebas



Como resultado de su ejecución se genera un informe en formato texto y HTML.

Figura 14. Informe de las pruebas

Para probar las funcionalidades presentadas por el cliente, se realizaron pruebas de aceptación a la aplicación. Estas pruebas se emplean en XP y se refieren a las pruebas funcionales de las HU realizadas por el equipo de desarrollo durante esta fase. El cliente especifica las vistas a probar cuando una HU se ha puesto en ejecución correctamente. Una HU puede tener una o

muchas pruebas de aceptación. Éstas son pruebas de caja negra que representa un resultado previo del sistema.

3.6.1.2 Pruebas de aceptación

Por cada iteración se traducen las Historias de Usuarios en pruebas de aceptación, con el objetivo de demostrar al cliente que el producto cumple con los requerimientos necesarios. Las pruebas de aceptación se muestran en el ***Anexo 4.***

A continuación se evidencia la igualdad en los resultados obtenidos para el índice de prioridad general en la evaluación de los proyectos que participaron en la validación de la tesis de maestría que propone el método MEPROI, defendida en el año 2010, como se puede apreciar en la *Figura 15* y los resultados arrojados por el sistema SEPI para los mismos juegos de datos, mostrados en la *Figura 16.*

Figura 15. Resultado de MEPROI Tesis de Maestría 2010

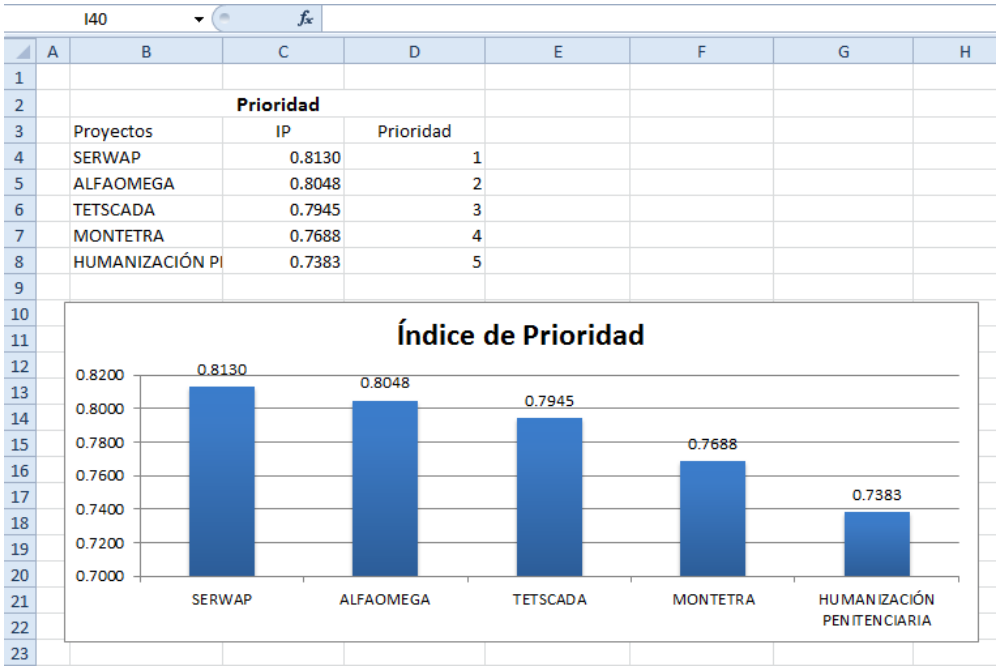
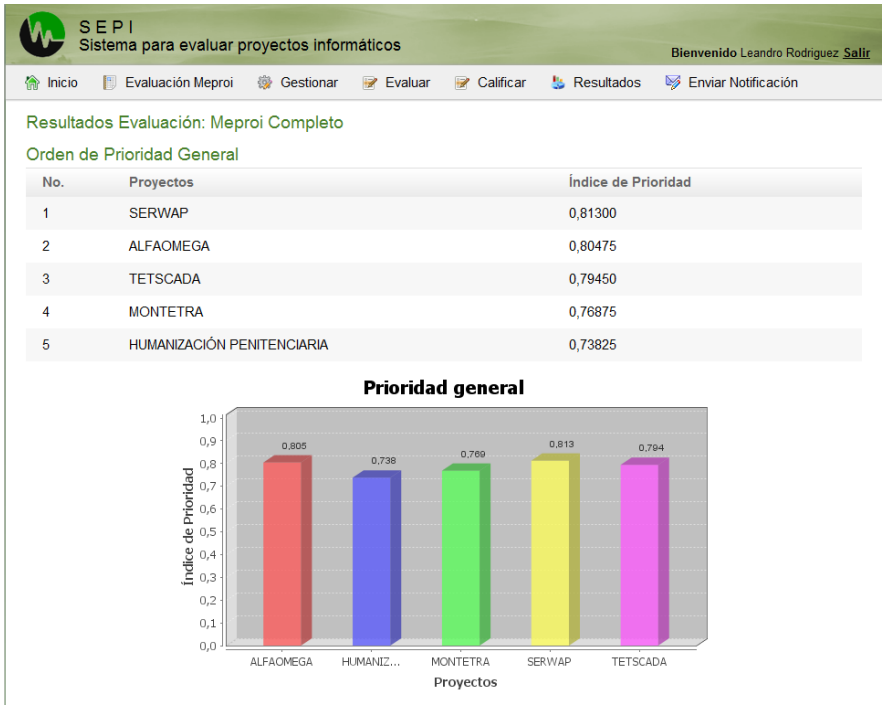


Figura 16. Resultado de MEPROI aplicando SEPI



3.7 Conclusiones parciales

- Se describieron los algoritmos de verificar concordancia y el cálculo del índice de prioridad de los proyectos.
- Se confeccionaron las tarjetas CRC y las tareas de la ingeniería las cuales permiten definir cada una de las actividades que estarán asociadas a las historias de usuario y que permitieron la implementación de éstas.
- Se prueba la solución informática con los niveles de pruebas de unidad y aceptación con resultados satisfactorios.

Conclusiones

Después de terminado el trabajo, se llegó a las conclusiones siguientes:

- El método MEPROI como herramienta de apoyo a la toma de decisiones constituye la base de la propuesta de solución y XP la metodología de desarrollo de software a utilizar.
- Se establecieron las tecnologías y herramientas para el desarrollo: Grails como marco de trabajo en su versión 2.0.3, PostgreSQL 9.1 como sistema gestor de base de datos, Apache Tomcat 7.0.27 como servidor de aplicaciones y el IntelliJ IDEA 11.0 como IDE de desarrollo.
- Se desarrolla la solución informática para evaluar proyectos y establecer un orden de prioridad que ayude a la toma de decisiones: SEPI.
- Se prueba la solución informática con los niveles de pruebas de unidad y aceptación con resultados satisfactorios.

Recomendaciones

Una vez concluido este trabajo se recomienda:

- Añadir la funcionalidad que permita eliminar el experto discordante como flujo alterno al no determinarse consenso en el trabajo de los expertos.
- Añadir otro mecanismo de autenticación que pueda ser configurable en el sistema.
- Añadir una funcionalidad que permita configurar el sistema de notificaciones.
- Utilizar el sistema para la aplicación del método MEPROI.

Referencias bibliográficas

1. Baca Urbina, Gabriel. *Evaluación de proyectos.*

2. Hernández, Rolando A. *Curso básico de Gestión de Proyectos.* 2007.

3. Sánchez Tamayo, Karina. Método para evaluar proyectos informáticos y establecer un orden de prioridad que ayude a la toma de decisiones. Ciudad de la Habana : s.n., 2010.

4. Buenas Tareas. *Buenas Tareas.* [Online] [Cited: enero 20, 2012.] http://www.buenastareas.com/ensayos/Gesti%C3%B3n-Del-Proyecto/3918343.html.

5. Labrada, Lisandra M. and Castro, Maylé (2009). Procedimiento para evaluar proyectos informáticos y establecer un orden de prioridades. Tesis de grado. Facultad 2. Universidad de las Ciencias Informáticas, Ciudad Habana, Cuba.

6. Siegel, S (1974). Estadística no paramétrica aplicada a las ciencias de la conducta. México D.F: Ed Trillas, págs. 262-273.

7. CMA, Colectivo de Matemática Aplicada (2008). Criterio de expertos: Método Delphi. Universidad de las Ciencias Informáticas, Ciudad Habana, Cuba.

8. IBM. [Online] [Cited: febrero 20, 2012.] http://www-01.ibm.com/software/awdtools/rup/.

9. Carvajal Riola, Jose Carlos. Metodologías Ágiles: Herramientas y modelo de desarrollo para aplicaciones Java EE como metodología empresarial. 2008

10. Extreme Programming:A gentle introduction. [Online] [Cited: febrero 24, 2012.] http://www.extremeprogramming.org/.

11. Beck, Kent. Extreme Programming Explained. 1999

12. Potencier, Fabien and Zaninotto, François. *Symfony la guía definitiva .*

13. Groovy and Grails. [Online] [Cited: febrero 26, 2012] http://www.springsource.com/developer/grails

14. Museo Histórico de Informática. [Online] [Cited: marzo 2, 2012] http://museo.informatica.uma.es/node/339

15. Grails . [Online] [Cited: marzo 10, 2012.] http://grails.org/plugin/mail.

16. Spring Security. [Online] [Cited: marzo 15, 2012] http://static.springsource.org/spring-security/site/index.html

17. JFreeChart. [Online][Cited: marzo 18, 2012] http://www.jfree.org/jfreechart/

18. IntelliJIDEA . [Online] [Cited: abril 2, 2012.] www.jetbrains.com/idea/.

19. Apache Tomcat. [Online] [Cited: marzo 30 , 2012.] http://tomcat.apache.org/.

20. PostgreSQL . [Online] [Cited: marzo 20 , 2012.] http://www.postgresql.org/about/

21. ANSI/IEEE Std 1471-2000. Recommended Practice for Architectural Description of Software-Intensive Systems.

22. Fowler, Martin. 2004. Inversion of Control Containers and the Dependency Injection pattern. [Online] Enero de 2004. [Cited: mayo 10, 2012] http://martinfowler.com/articles/injection.html

23. Adictos al Trabajo. *Adictos al Trabajo.* [Online] [Cited: mayo 20, 2012.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp

24. Gutiérrez, J. J., et al. *Pruebas del sistema en Programación Extrema.* Departmento de Lenguajes y Sistemas Informáticos : s.n.

Bibliografía

1. Sánchez Tamayo, Karina. Método para evaluar proyectos informáticos y establecer un orden de prioridad que ayude a la toma de decisiones. Ciudad de la Habana : s.n., 2010.

2. Grails. [Online] http://www.grails.org/Tutorials.

3. Grails. [Online] http://observatoriodegrails.com/2010/09/30/relaciones-mn-multiples-con-grails-ingles/.

4. Grails. [Online] http://grails.codehaus.org/.

5. Spring Source. [Online] http://static.springsource.org/spring-security/site/index.html.

6. Groovy and Grails Recipes. [book auth.] Bashar Abdul-Jawad. s.l. : Steve Anglin, Tom Welsh, 2009.

7. Brito, Nacho. *Manual de Desarrollo Web con Grails.* 2009.

8. Christopher M. Judd, Joseph Faisal Nusairat, and James Shingler. *Beginning Groovy and Grails.* 2008.

9. The Definitive Guide to Grails.Second Edition. [book auth.] Jeff Brown Graeme Rocher. 2009.

10. Rudolph, Jason. Getting Started with Grails. s.l. : C4Media, Publisher of InfoQ.com., 2006.

11. Subramaniam, Venkat. Programming Groovy, Dynamic Productivity for the Java Developer. Raleigh, North Carolina Dallas, Texas : s.n., 2008.

12. Glen Smith, Peter Ledbrook. *Grails in Action.* s.l. : Manning Publications Co, 2009.

13. Dierk Konig, Andrew Glover. *Groovy in Action.* s.l. : Manning Publications Co, 2007.

14. Spring Security. [Online] http://static.springsource.org/spring-security/site/index.html

15. JFreeChart. *JFreeChart.* [Online] http://www.jfree.org/jfreechart/

16. XP Extreme Programming [Online] http://s3.amazonaws.com/engrade-myfiles/4097134021351167/clase7b-Metodologia-XP.pdf

17. Carvajal Riola, Jose Carlos. Metodologías Ágiles: Herramientas y modelo de desarrollo para aplicaciones Java EE como metodología empresarial. 2008

18. Joskowicz, José. Reglas y Prácticas de Extreme Programming. 2008

19. Echeverry Tobón, Luis M. and Delgado Carmona, Luz Elena. Caso Práctico de la Metodología Ágil al desarrollo de software. 2007

20. Desarrollo Ágil con SCRUM[Online] http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:sg07.p02.scrum.pdf

21. Extreme Programming: A gentle introduction.[Online] [Cited: febrero 24, 2012.] http://www.extremeprogramming.org/.

22. Beck, Kent. Extreme Programming Explained. 1999

23. Grails . [Online] http://grails.org/plugins

24. Java. [Online] http://docs.oracle.com/javase/tutorial/

25. Groovy [Online] http://groovy.codehaus.org/

26. Escuela de Groovy. [Online] http://www.escueladegroovy.com/

27. PostgreSQL. [Online] http://www.postgresql.org/about/.

28. IntelliJIDEA. [Online] www.jetbrains.com/idea/.

29. Apache Tomcat. [Online] http://tomcat.apache.org/.

30. Fowler, Martin. 2004. Inversion of Control Containers and the Dependency Injection pattern. [Online]. http://martinfowler.com/articles/injection.html.

Anexos

Anexo 1. Historias de Usuario

Iteración 1

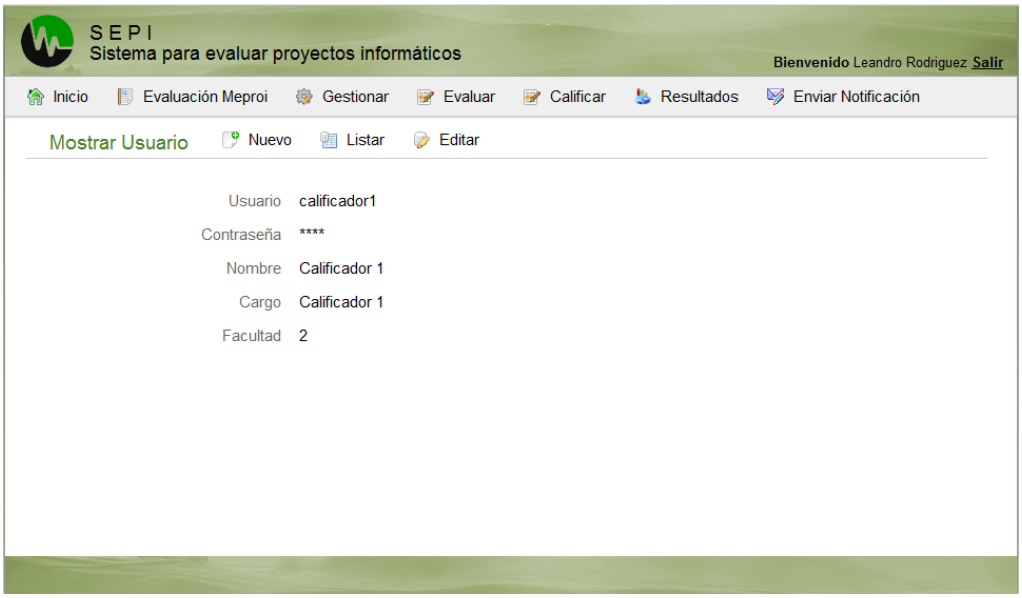
Historia de Usuario	
Número: HU-2	**Nombre Historia de Usuario**: Gestionar usuario.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada**: Primera.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: El analista tendrá los permisos para listar, crear, modificar, buscar y eliminar usuarios que interactúan con el sistema.	
Observaciones:	
Prototipo de interfaz: 	

Tabla 31. HU-2: Gestionar usuario

Historia de Usuario	
Número: HU-3	**Nombre Historia de Usuario**: Gestionar proyecto.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada**: Primera.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: El analista tendrá los permisos para listar, crear, modificar, buscar y eliminar proyectos.	
Observaciones:	

Prototipo de Interfaz:

Tabla 32. HU-3: Gestionar proyecto

Historia de Usuario	
Número: HU-4	**Nombre Historia de Usuario**: Gestionar criterio.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez. .	**Iteración Asignada**: Primera.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: El analista tendrá los permisos para listar, crear, modificar, buscar y eliminar los criterios según los cuales serán evaluados los proyectos.	
Observaciones:	

Prototipo de Interfaz:

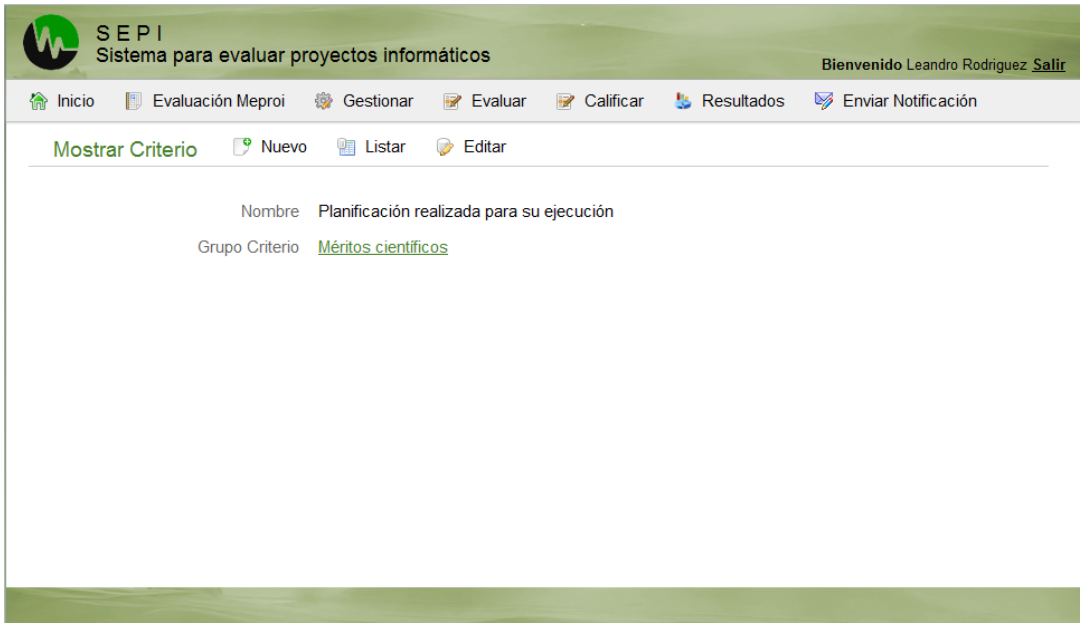


Tabla 33. HU-4: Gestionar criterio

Historia de Usuario	
Número: HU-5	**Nombre Historia de Usuario**: Gestionar grupo criterio.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada**: Primera.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: El analista tendrá los permisos para listar, crear, modificar, buscar y eliminar los grupos de criterios a los cuales pertenecen cada uno de los criterios.	
Observaciones:	

Prototipo de Interfaz:

Tabla 34. HU-5: Gestionar grupo criterio

Historia de Usuario	
Número: HU-6	**Nombre Historia de Usuario**: Gestionar evaluación Meproi.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada**: Primera.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: El analista tendrá los permisos para listar, crear, modificar, buscar y eliminar las evaluaciones Meproi.	
Observaciones:	

Prototipo de Interfaz:

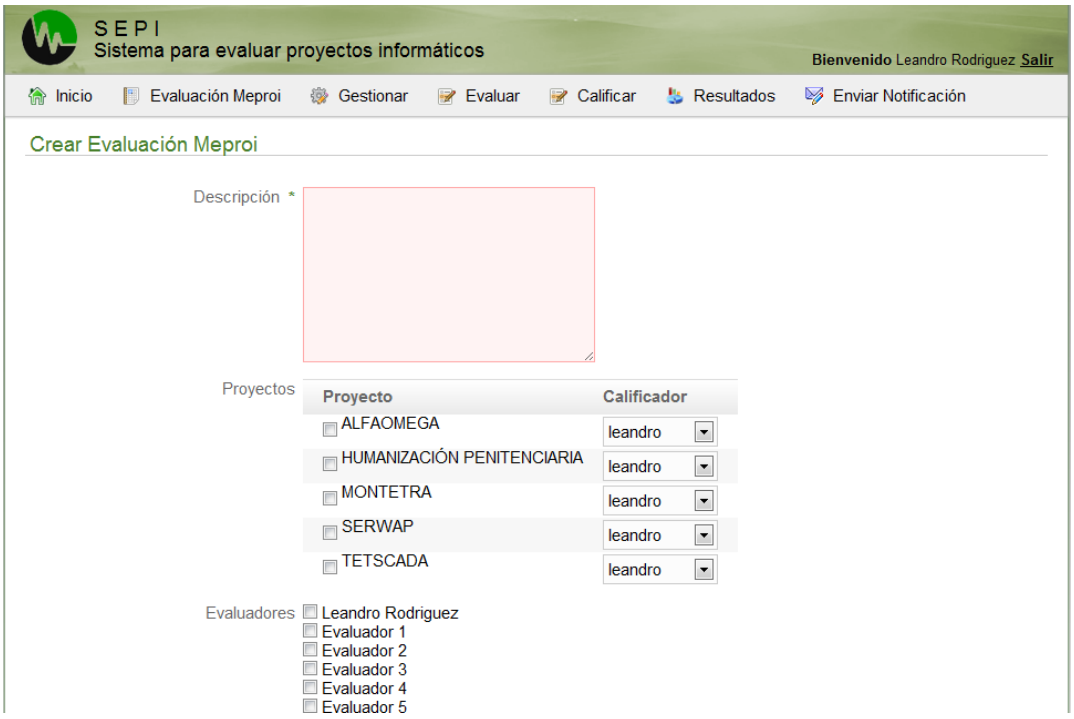


Tabla 35. HU-6: Gestionar evaluación Meproi

Iteración 2

Historia de Usuario	
Número: HU-7	**Nombre Historia de Usuario**: Realizar evaluación peso-criterios.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada**: Segunda.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: La aplicación permitirá a cada uno de los evaluadores realizar la evaluación correspondiente mediante un formulario, en el cual se listarán los criterios. El sistema debe también ser capaz de guardar esta evaluación en la base de datos.	
Observaciones:	

Prototipo de Interfaz:

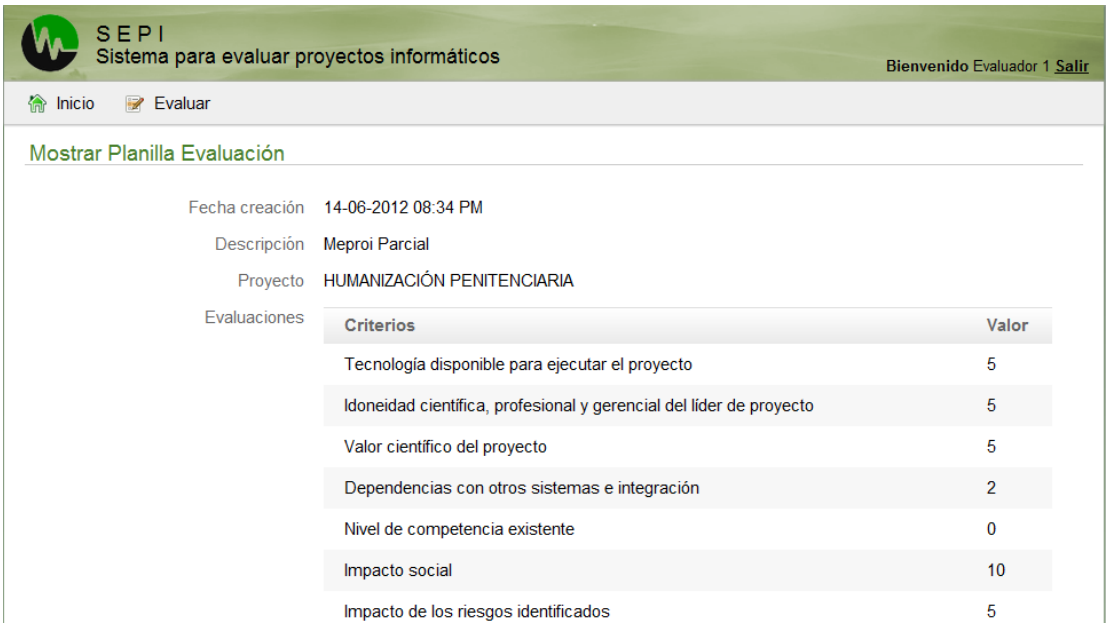


Tabla 36. HU-7: Realizar evaluación peso-criterios

Historia de Usuario	
Número: HU-9	**Nombre Historia de Usuario:** Verificar concordancia expertos
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada**: Segunda.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: La aplicación realizará el procedimiento matemático mediante el cual se verifica que exista concordancia en el criterio de los expertos.	
Observaciones:	
Prototipo de Interfaz: No aplica.	

Tabla 37. HU-9: Verificar concordancia expertos

Historia de Usuario	
Número: HU-8	**Nombre Historia de Usuario**: Realizar calificación criterios.
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada**: Segunda.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: La aplicación permitirá al calificador seleccionado realizar la evaluación correspondiente	

mediante un formulario, en el cual se listarán los criterios. El sistema debe también ser capaz de guardar esta evaluación en la base de datos.
Observaciones:
Prototipo de Interfaz: 

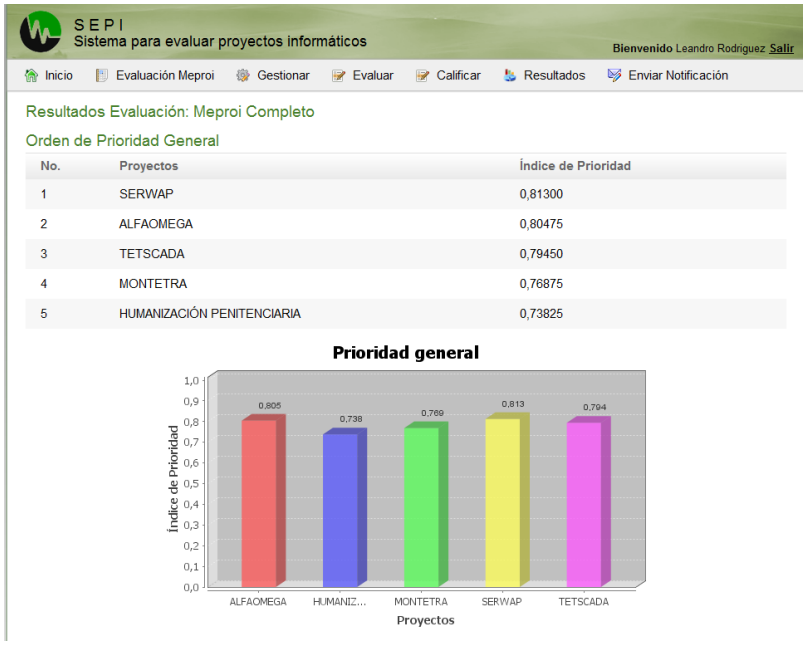
Tabla 38. HU-8: Realizar calificación criterios

Historia de Usuario	
Número: HU-10	**Nombre Historia de Usuario:** Calcular índice de prioridad
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada**: Segunda.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: La aplicación realizará el procedimiento matemático mediante el cual se calcula el índice de prioridad para cada uno de los proyectos de la evaluación.	
Observaciones:	
Prototipo de Interfaz: No aplica.	

Tabla 39. HU-10: Calcular índice de prioridad

Iteración 3

Historia de Usuario

Número: HU-11	**Nombre Historia de Usuario:** Graficar resultados evaluación
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada**: Tercera.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: La aplicación será capaz de representar de forma gráfica los resultados de la evaluación Meproi.	
Observaciones:	
Prototipo de Interfaz: 	

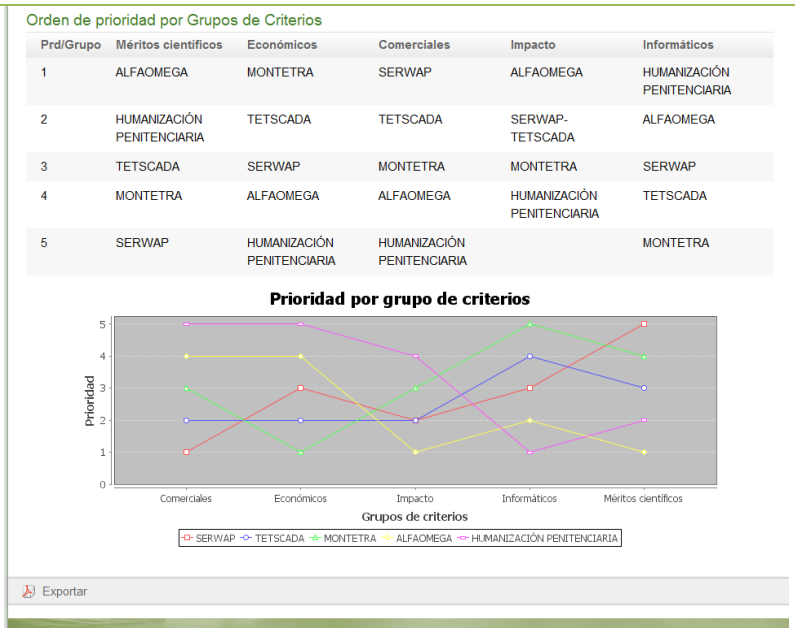


Tabla 40. HU-11: Graficar resultados evaluación

Historia de Usuario	
Número: HU-12	**Nombre Historia de Usuario**: Enviar notificaciones vía correo electrónico
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Leandro José Rodríguez Hdez.	**Iteración Asignada**: Tercera.
Prioridad en el negocio: Muy alta.	**Puntos Estimados**: 1
Descripción: La aplicación permitirá realizar notificaciones por correo electrónico por parte de los analistas del sistema.	
Observaciones:	

Prototipo de Interfaz:

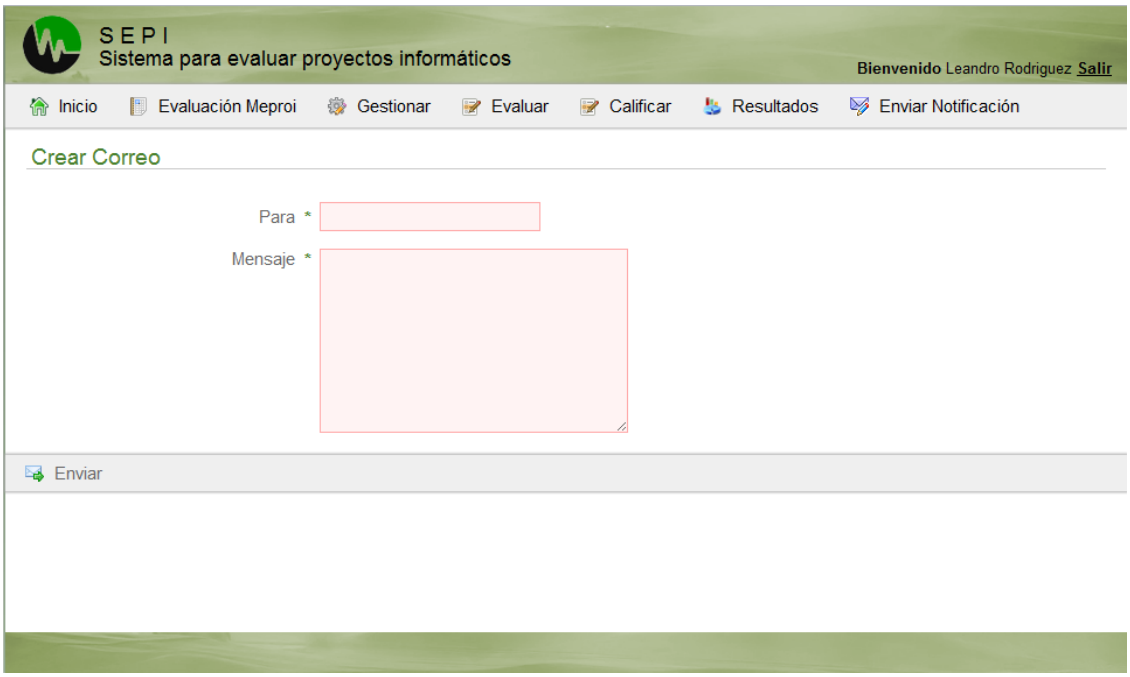


Tabla 41. HU-12: Enviar notificaciones vía correo electrónico