UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

REDISEÑO E IMPLEMENTACIÓN DEL MÓDULO DE RECURSOS HUMANOS DE LA HERRAMIENTA GESPRO 13.05

Autor: José Antonio Pellicer Allen

Tutores: Msc. Surayne Torres López

Ing. Rosel Sosa González

La Habana 2014

Pensamiento

Pude haber sido el hombre del ayer, pero eso fue ayer, hoy estoy aquí en el nombre del mañana, un mañana que a diferencia de sus mañanas, será construido sobre la grandeza del ayer, aunque obviamente sucederá hoy.

Declaración de autoría

Declaro ser el autor del trabajo de diploma titulado "Rediseño e implementación del módulo de Recursos Humanos de la herramienta GESPRO 13.05" y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Autor:

_____

José Antonio Pellicer Allen

Tutores:

_____ _____

Surayne Torres López Rosel Sosa González

Datos de contacto

MSc. Surayne Torres López

Graduada de Ingeniería en Ciencias Informáticas en el 2008. Profesora Instructora con 6 años de experiencia como docente y especialista en el área de Gestión de Proyectos. Máster en Gestión de Proyectos Informáticos.

Correo: storres@uci.cu

Ing. Rosel Sosa González

Graduado de Ingeniería en Ciencias Informáticas en el 2013.

Correo: rsosag@uci.cu

Agradecimientos

Le agradezco:

- *A mi tutor Rosel por toda la ayuda apoyo y confianza, por las horas que dedicó a la realización y revisión de este trabajo, por dejar a un lado lo demás y prestarme su atención y por su amistad.*
- *A mi tutora Surayne porque estuvo siempre dispuesta a ayudarme en todo, por la confianza que depositó en mí, por los consejos y sugerencias.*
- *Al profesor Reynaldo porque gracias a él obtuve el tema de este trabajo, necesario para graduarme como ingeniero, por la ayuda que me brindó en aquel momento, que a pesar de no conocerme, confió en mí.*
- *A mi profesora de MIC Anisleiby, jefa de tribunal, por la ayuda prestada y por los consejos ofrecidos.*
- *A Félix, oponente del trabajo, porque gracias a sus recomendaciones el trabajo ha adquirido la calidad necesaria.*
- *A los profesores del laboratorio por guiarme y ayudarme cuando lo he necesitado.*
- *A los profesores vinculados a mi formación.*
- *A mi gran amiga Miriela por soportarme en todo momento y aconsejarme siempre y por la gran ayuda que me brindó en gran parte de la confección del documento. Besos para ti.*
- *A mi socio el Vity por la amistad incondicional que lo caracteriza, por toda la preocupación que ha tenido por mí desde la adolescencia y hasta el día de hoy.*
- *A mis amigos y compañeros de aula con los que he cursado estos 5 años, a Rafael por que ha sido como un hermano para mí en las buenas y las malas, a Joaquín, Pedro, White, Anisley, Judith, Luis, Yariel, Reynol y todos los demás.*
- *A mis amigos y compañeros de guerrillas por todos los buenos momentos, las tremendas fiestas, peñas, conciertos y festivales de ROCK N' ROLL que hemos tenido la oportunidad de asistir y aunque ya muchos no estén en la escuela siempre estarán en mis recuerdos.*
- *A Daniela por toda la ayuda que me brindó, que a pesar de estar en Holguín, siempre estuvo dispuesta para ayudarme en todo lo posible.*
- *A mi familia por todo el cariño y amor que siempre han tenido por mí y por alentarme y apoyarme en todo momento.*
- *A mis amigos del apartamento que a pesar de conocernos en este último año su amistad ha sido incondicional.*

A todos muchas gracias…

Dedicatoria

Dedico este trabajo:

A mi madre por ser la mejor madre del mundo, por estar siempre a mi lado, por brindarme su confianza, cariño y amor, por todos aquellos consejos y el apoyo incondicional. Todo te lo debo a ti mi vieja.

A mi padre por ser mi ejemplo a seguir, por su apoyo y consejos, por guiarme en todo el transcurso de mi vida.

A mi hermano José Manuel por la confianza que tuvo en mí, por brindarme el apoyo y el cariño que solo los hermanos podemos ofrecer y por estar siempre presente en todo momento.

A mis abuelos Hilda, Roberto y Sara por todo ese amor y cariño que sienten por mí y que me ha llenado de felicidad por toda la vida.

A mis tías Lourdes, Sara, Bertha por ser mis segundas madres y darme todo su amor y apoyo.

A toda mi familia por estar presentes en los momentos buenos y malos, por apoyarme y alentarme.

En especial quiero dedicar este trabajo al gran hombre, padre, tío, hermano e hijo que fue, a la memoria de mi querido tío Kiko que a pesar de no poder verme en este momento estoy seguro de que estría muy orgulloso de mí. Gracias tío por todo lo que hiciste por mí y por tratarme como un hijo más, para ti dedico este trabajo por todo el amor que siempre me tuviste y por confiar en mí.

Resumen

El uso de herramientas informáticas para la gestión de proyectos se ha ido generalizando con el objetivo de facilitar el trabajo de los especialistas vinculados a esta actividad. Entre estas herramientas se encuentra la plataforma XEDRO GESPRO, aplicación para la Gestión de Proyectos desarrollada por la Universidad de las Ciencias Informáticas (UCI), que tiene entre sus funciones la gestión de Recursos Humanos.

La versión más reciente de esta plataforma, GESPRO 13.05, no posee funcionalidades asociadas a la gestión de las actividades de formación como parte de la gestión de Recursos Humanos, siendo uno de los procesos claves en la gestión de proyectos, resolverlos constituye un mérito de GESPRO frente a otras herramientas de gestión de proyectos.

Se propone rediseñar e implementar el módulo de Recursos Humanos para facilitar la gestión de los mismos en la plataforma XEDRO GESPRO, el que permitirá gestionar actividades de formación y proponer un plan de formación, necesario para aumentar el nivel de competencia del equipo de trabajo, según las necesidades de los proyectos.

Para llevar a cabo el trabajo se confecciona un diseño que cumple con los parámetros establecidos por la arquitectura y se emplean las herramientas y tecnologías seleccionadas para la implementación. Además, se realizan pruebas para la validación de las funcionalidades, asegurando que el módulo realizado cumpla con las normas vigentes para el desarrollo.

Palabras claves: Plan de formación, gestión de Recursos Humanos, GESPRO.

Índice

Índice de figuras

Índice de Tablas

Introducción

El avance de las tecnologías de la información y las telecomunicaciones en el nuevo milenio y su auge cada vez más creciente, han tenido un papel preponderante en la automatización de la mayor parte de los procesos sociales, empresariales y políticos, por consecuencia el desarrollo de la industria del *software* ha sido un factor determinante en dicho progreso. Actualmente existe una gran competencia en el desarrollo y comercialización de *software*, por lo que las empresas inmersas en esta esfera concentran sus esfuerzos en crear aplicaciones con mayor calidad en correspondencia con las necesidades del cliente.

Una de las disciplinas que permite alcanzar un alto nivel en la calidad de *software* es la gestión de proyectos, principalmente porque se encarga de organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo y costos definidos (Achell, 2010). El uso de herramientas para la gestión de proyectos se ha ido generalizando al paso del tiempo con el objetivo de facilitar el trabajo de los especialistas (Laboratorios de Gestión de Proyectos, 2012).

Las Instituciones enmarcadas en el desarrollo y producción de *software* han dedicado gran esfuerzo a la aplicación o creación de herramientas para la gestión de proyecto, como *Serena Software*, *Project.net*, *OnePoint Software*, *Oracle* e incluso *Microsoft* en el plano internacional (Gartner, 2013). Cuba, con el objetivo de no quedarse al margen de las tendencias tecnológicas mundiales ha potenciado el impulso de empresas de esta índole, dentro de las que se encuentran centros como el Desarrollo de Aplicaciones, Tecnologías y Sistemas (*DATYS*).

Una de las instituciones antes mencionadas es la Universidad de las Ciencias Informáticas (*UCI*) la cual es un centro científico-productivo que cuenta, dentro de su infraestructura productiva, con el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (*CDAE*). Este centro se ha adentrado en el campo correspondiente a la gestión de proyecto, un ejemplo representativo lo constituye el sistema GESPRO, que actualmente se encuentra en explotación su versión 13.05, el cual es una Suite de herramientas con funcionalidades asociadas a áreas de la gestión de proyectos como el alcance y el tiempo, así como el control y seguimiento de los proyectos de los centros de desarrollo y la gestión de los Recursos Humanos.

Esta Suite se presenta en un modelo de negocio basado en servicios que combinan el uso de una solución informática para la dirección integrada de proyectos y un sistema de formación especializada en gestión de los mismos. Esta combinación posibilita tanto la informatización de las organizaciones como la mejora integral de los procesos de planificación, control y seguimiento de proyectos (Torres Quiñones, 2014).

GESPRO 13.05, con el objetivo de mejorar los procesos de gestión de proyectos, posee un módulo para gestionar los Recursos Humanos, el cual almacena datos personales de los usuarios que pertenecen a proyectos y los roles desempeñados por los mismos en cada uno de ellos, dicho módulo guarda información relacionada con las competencias laborales y la evaluación de las mismas (Alfonso, 2012). En entrevistas realizadas a especialistas del proyecto GESPRO y durante la revisión de las funcionalidades del propio módulo en el sistema instalado al centro CDAE, se determinó que el mismo no presenta funcionalidades asociadas a la gestión de los planes de formación y la propuesta de los mismos según la necesidad de cada proyecto. Los planes de formación son partes del proceso de desarrollo de los Recursos Humanos destinados a la formación continua del personal (PMI, 2013).

Durante la revisión de investigaciones realizadas como parte del sistema de formación especializada se identificó, en una tesis de maestría, un algoritmo para proponer planes de formación. El mismo fue implementado en un módulo, el cual pretendía completar las funcionalidades del sistema para la gestión de Recursos Humanos (Cuza García, 2013). Sin embargo, este módulo aislado fue desarrollado utilizando una versión anterior del núcleo base Redmine. También presenta el inconveniente de duplicar funcionalidades existentes en el módulo Recursos Humanos, además de no utilizar los modelos definidos en la herramienta GESPRO 13.05 para acceder a los datos de la misma. Estas insuficiencias no permitieron la integración de dicho módulo a la Suite de herramientas y por tal motivo la gestión de Recursos Humanos del sistema XEDRO GESPRO no presenta funcionalidades asociadas a la formación de sus usuarios.

Los impedimentos antes mencionados y la necesidad de integrar funcionalidades aisladas concretan la base para un cambio en la estructura del módulo de Recursos Humanos que permita mejorar los procesos de gestión de los mismos.

Una vez analizada la situación problemática, se define como **problema de la investigación** que las insuficiencias en el diseño del módulo de Recursos Humanos afecta la propuesta de planes de formación basados en competencias laborales en la herramienta GESPRO 13.05.

Para darle solución al problema de la investigación antes expuesto se enmarca como **objeto de estudio** el diseño del módulo de Recursos Humanos de GESPRO 13.05.

Como **objetivo general** del presente trabajo de diploma se propone: rediseñar e implementar el módulo de Recursos Humanos de la herramienta GESPRO 13.05 para que permita la propuesta de planes de formación basados en competencias laborales.

Derivándose como **Campo de acción** los componentes del módulo de Recursos Humanos de la herramienta GESPRO 13.05.

Para cumplimentar el objetivo general se plantean como **tareas de la investigación** las siguientes:

1. Análisis de los principales conceptos asociados a la investigación.
2. Análisis de las características de diseño e implementación del módulo de Recursos Humanos de GESPRO 13.05.
3. Análisis de las características de diseño e implementación del módulo de creación de planes de formación basado en competencias laborales.
4. Estudio de la metodología de desarrollo de *software*, herramientas y tecnologías para el rediseño e implementación del módulo.
5. Especificación de los requerimientos funcionales y no funcionales del módulo.
6. Diseño para el nuevo módulo de Recursos Humanos para la herramienta GESPRO 13.05.
7. Implementación del nuevo módulo de Recursos Humanos para la herramienta GESPRO 13.05.
8. Validación mediante las pruebas de caja negra a las funcionalidades del nuevo módulo de Recursos Humanos para su futura utilización en GESPRO 13.05.

Los **Métodos Científicos** utilizados son:

Métodos teóricos:

- **Analítico-Sintético:** se hace uso de este método para analizar a través de diversas fuentes bibliográficas los elementos teóricos asociados al rediseño de sistemas.

- **Modelación:** la utilización de este método permite la confección de los diagramas correspondientes al diseño del *software*, con el fin de facilitar la comprensión del funcionamiento del módulo.

Métodos empíricos:

- **Entrevista:** se hace uso de este método para interactuar con personal capacitado, con el fin de obtener información e ideas que contribuyan al desarrollo de la investigación.

- **Observación:** Observar el funcionamiento de los módulos para la gestión de Recursos Humanos y creación de planes de formación basados en competencias laborales existentes para de esta manera comprender las funcionalidades básicas que debe de tener el nuevo componente a desarrollar, así como ver las posibilidades para mejorarlo.

El presente trabajo de diploma consta de tres capítulos estructurados de la siguiente manera:

Capítulo 1: Fundamentación teórica: se enfoca fundamentalmente en conceptos y elementos teóricos relacionados con la investigación a realizar, la exposición de las principales características del módulo que interviene en el desarrollo del trabajo, así como un estudio de metodologías de desarrollo de *software*, herramientas y tecnologías que se utilizarán para el desarrollo del presente trabajo investigativo.

Capítulo 2: Análisis y diseño: orientado principalmente al análisis de las principales características de la solución a proponer, se fundamentará la fase de análisis y diseño, en la cual se expondrán los requerimientos tanto funcionales como no funcionales que debe tener el módulo a desarrollar, además de mostrar una representación de la ingeniería del *software*.

Capítulo 3: Implementación y prueba: se expone la fase de desarrollo teniendo en cuenta las especificaciones obtenidas en los diseños de las clases, además de las pruebas a las que serán sometidas las funcionalidades y los resultados obtenidos al realizar las mismas con las cuales se comprobará la calidad y efectividad del producto final.

Capítulo 1: Fundamentación teórica

Introducción

En el presente capítulo se realizará una investigación para el análisis de los principales conceptos relacionados a la investigación. Se exponen las principales características presentes en los sistemas involucrados en el proceso investigativo y se explica la necesidad del rediseño. Se define la metodología de desarrollo de *software*, las herramientas y tecnologías que se utilizarán para dar cumplimiento al objetivo del presente trabajo.

1.1 Conceptos asociados a la investigación

A continuación se muestran los principales conceptos asociados con el objeto de estudio y desarrollo de la propuesta y serán el punto de partida en el trayecto hacia la meta a seguir por el presente trabajo.

1.1.1 Reingeniería

En la bibliografía se pueden encontrar diferentes conceptos de reingeniería. Algunos de los más relacionados con la investigación se exponen a continuación. Estos conceptos sirven de orientación para la propuesta de la solución.

La Reingeniería consiste en rediseñar los procesos, de manera que estos estén fragmentados. Propiamente hablando se pudiera definir como la revisión fundamental y el rediseño radical de procesos para alcanzar mejoras espectaculares en medidas críticas y actuales de rendimiento, tales como costo, calidad, servicio y rapidez (Juárez García , et al., 2008).

El término se refiere al rediseño rápido y radical de los procesos estratégicos de valor agregado y de los sistemas, las políticas y las estructuras organizacionales que los sustentan para optimizar los flujos del trabajo y la productividad de una organización (Manganelli, et al., 1995).

De acuerdo con Hammer y Stanton, es repensar de manera fundamental los procesos de negocios y rediseñarlos radicalmente, con el fin de obtener dramáticos logros en el desempeño. Los factores clave del concepto son: la orientación hacia los procesos, el cambio radical y la gran magnitud de los resultados esperados (Hammer, et al., 1997).

1.1.2 Rediseño

En ocasiones se confunden los conceptos de "reingeniería" y "rediseño", se emplean como sinónimos, pero no lo son. El rediseño de procesos, no es tan radical como la reingeniería; puede, por ejemplo, aplicarse a una parte del proceso de negocio y tiene como objetivo mejorar el grado de competitividad a través de técnicas de optimización de procesos. El mayor impacto de un rediseño se tiene si el análisis comienza con los eventos generados por los clientes y los resultados que llegan a ellos, por ejemplo solicitudes, pedidos, pagos, reclamos, entre otros. Las dimensiones de optimización en el rediseño son: reducción de los tiempos de ciclo, mejoramiento de la calidad de los productos y servicios y reducción de costos (Hitpass Heyl, 2011).

El rediseño establece los cambios que deberán efectuarse en la situación actual y determina cómo se ejecutarán los nuevos procesos. Es la fase más importante, ya que se definirán las nuevas formas de operar y su desempeño.

El rediseño influye en los siguientes aspectos:

Estructural: Cambio en el proceso mismo (cambian las operaciones y se eliminan duplicidades).

Productividad: Análisis de ciclo y costeo de actividades.

Responsabilidades: Se modifica la asignación de responsabilidad (personal, centralizar o descentralizar responsabilidades).

Integración: Mejorar el grado de integración entre la capa de la estrategia, operacional (procesos) y tecnología (producción).

Incorporación de tecnología: Automatización de procesos, aplicación de tecnologías móviles, integración de sistemas, y más.

1.1.3 Mejora continua de procesos

El concepto de mejora continua está limitado a cambios pequeños como reglas de negocio, procedimientos locales, redistribución del volumen de trabajo, simplificación de formularios, entre otros. Además, se puede entender como el solo monitoreo del rendimiento de los procesos a través de indicadores de ciclo u otros y comenzar iniciativas de mejora cuando se detectan desviaciones al comportamiento esperado (Hitpass Heyl, 2011).

En la siguiente tabla se exponen las principales características que diferencian estos enfoques en la mejora de procesos descritos por Bernhard Hitpass Heyl.

Tabla 1 Principales características que diferencian los tres enfoques principales de mejora de procesos (Hitpass Heyl, 2011).

Característica	Reingeniería	Rediseño	Mejora
Enfoque	Proceso nuevo	Restructuración	Mejora evolutiva
Punto de partida	Proceso existente	Proceso existente	Proceso existente
Objetivo del cambio	Cambio radical, satisfacción del cliente	Rediseño de una parte del proceso	Actualización, eficiencia o satisfacción del cliente
Tipo de cambio	Radical	Estructural	Incremental
Periodicidad del cambio	Descontinuado	Intervalos intermedios	Continuo
Organización del cambio	Proyecto	Proyecto o grupo de trabajo	Dentro de operaciones
Impulsor del cambio	Directorio	Dueño de proceso	Cualquier actor
Impacto del cambio	Transversal	Proceso, subproceso	Dentro de un Subproceso
	Cultural	Cultural	Cognitivo
	Procesal	Procesal	Procedimiento, regla de negocio
	Estructural	Estructural	Costo, calidad, tiempo

Una vez analizados los anteriores conceptos se puede decir que si los cambios propuestos por la mejora continua impactan sobre la estructura de los procesos, traspasan los límites de responsabilidad del área, impactan sobre la tecnología, o bien requieren de recursos adicionales, la propuesta de mejora pasa a ser un proyecto de rediseño. De igual forma si un proyecto de rediseño pone en duda la estructura de responsabilidades o traspasa las fronteras de un área de negocio, pasa a ser un proyecto de reingeniería.

1.1.4 Reutilización

Otro término involucrado en la investigación es la reutilización que según la Real Academia Española, “Reutilizar” es utilizar algo, bien con la función que desempeñaba anteriormente o con otros fines (Real Academia Española, 2001).

Peter Freeman expone que el propósito de la reutilización es mejorar la eficiencia, la productividad y la calidad del desarrollo de *software*. Así la reutilización puede definirse como “cualquier procedimiento que produce o ayuda a producir un sistema mediante el nuevo uso de algún elemento procedente de un esfuerzo de desarrollo anterior” (Freeman, 1987).

Por otra parte, se define como “La utilización de elementos *software* existentes durante la construcción de un nuevo sistema software” (Hutchison, 2008).

La reutilización ha sido, y sigue siendo, uno de los principales temas de investigación en el campo de la Ingeniería del *Software*, citándose frecuentemente como una de las principales técnicas para incrementar la productividad de los desarrolladores de *software*.

Es considerada como una buena práctica en el proceso de desarrollo de *software* debido a que permite utilizar un esfuerzo anterior con el objetivo de crear aplicaciones de mayor rendimiento en cuanto a la eficiencia, productividad y calidad.

1.1.5 Gestión de proyectos

Cuando se emplea el concepto “gestión” para designar el conjunto de actividades que se realizan para desarrollar un proyecto desde su concepción hasta su introducción en la práctica social, se debe definir de la forma siguiente:

Gestión de proyectos: Es la aplicación de conocimiento, habilidades y técnicas de planificación, organización y control para optimizar las actividades de un proyecto y obtener los objetivos propuestos, cumpliendo con el tiempo establecido, el presupuesto previsto y la calidad requerida (PMI, 2013).

En la gestión de un proyecto intervienen cuatro factores fundamentales que se describen a continuación: control que se mantiene sobre la ejecución del proyecto, factores que pueden llevar a la necesidad de cancelación del proyecto, comercialización de los resultados que se obtendrán con el proyecto y uso de los Recursos Humanos que intervienen en la ejecución del proyecto (Hernández León, 2009).

1.1.6 Gestión de Recursos Humanos

Para introducir el término de gestión de Recursos Humanos primero se debe indagar acerca de los propios Recursos Humanos, los cuales se entienden como las personas que ingresan, permanecen y participan en la organización, en cualquier nivel jerárquico o tarea (Chiavenato, 2007).

Una vez analizada la definición anterior se puede definir a la gestión de Recursos Humanos como un conjunto de actividades coordinadas que ponen en funcionamiento, desarrollan y movilizan a las personas que una organización necesita para realizar sus objetivos (Gómez Mejía, et al., 2008).

Este conjunto de actividades dirige y controla una organización lo que permite materializar la política laboral que se aplica con la participación activa y efectiva de los trabajadores en la planificación, organización dirección, control y evaluación de los Recursos Humanos, que determinan o inciden en el desempeño de la organización (PMI, 2013).

1.1.7 Plan de formación basados en competencias laborales

Se expresa como plan de formación al conjunto de actividades cuyo propósito es mejorar el rendimiento presente o futuro, aumentando la capacidad a través de la mejora de sus conocimientos (Valenciano, 2009).

Armando Cuesta expone el término como características subyacentes en las personas, asociadas a la experiencia, que como tendencia están causalmente relacionadas con actuaciones exitosas en un puesto de trabajo, contextualizado en determinada cultura organizacional (Cuesta Santos, 2010).

El plan de formación o capacitación se fundamenta en disminuir las diferencias entre las competencias presentadas por la persona y las requeridas por el puesto, en los casos centrados a competencias laborales y en resolver las necesidades identificadas en la fase inicial para el resto de los procedimientos (Cuza García, 2013).

1.2 Características del sistema

Como problemática de la investigación se delimitó que el módulo creado para gestionar la información relacionada a las actividades de formación presenta insuficiencias en el diseño e implementación, debido a esto se ve afectada la integración del mismo al módulo Recursos Humanos. Rediseñar este último está dada por la necesidad de generar planes de formación basados en competencias laborales.

1.2.1 Módulo de Recursos Humanos

GESPRO en su versión 13.05 es una plataforma extensible que está formada a partir de la integración de más de 18 herramientas libres, comercializadas bajo licencia GPL[1]. Se ha aplicado con buenos resultados en la red de centros de la Universidad de las Ciencias Informáticas, que incluye varios centros de desarrollo de *software*. Actualmente es utilizado por más de 6000 usuarios que gestionan actividades de más de 150 proyectos entre los que se encuentran proyectos para la informatización nacional como para la exportación. Este sistema ha sido desarrollado utilizando los modelos de líneas de productos de *software* como modelo industrial de desarrollo (Piñero, 2010).

El sistema cuenta con un módulo para la gestión de Recursos Humanos encargado de administrar y dirigir el personal de trabajo de los proyectos de desarrollo, dicho módulo fue diseñado e implementado bajo las especificaciones técnicas de la herramienta antes mencionada, el mismo está dividido en dos niveles, a nivel de centro y de proyecto, presentando funcionalidades que facilitan la gestión de los Recursos Humanos.

- **Nivel de centro**
  - gestionar competencias, permite crear, modificar y eliminar competencias.
  - gestionar dimensiones la cual añade, modifica y elimina las dimensiones asociadas a una competencia.
  - mostrar los roles asociados a competencias, asigna y elimina las competencias asociadas a los mismos.
  - mostrar la evaluación del usuario en una competencia según eficacia, eficiencia, efectividad y la consistencia del conjunto utilizando la función de similitud de textos cosenos, además de la evaluación del sistema, esta última mediante algoritmos, y la evaluación del experto la que puede ser modificada, por último da la opción de eliminar la competencia asociada al usuario.
  - mostrar competencias por puesto de trabajo.
  - mostrar la evaluación del desempeño de cada usuario.
  - mostrar un resumen de asistencias por usuario.
- **Nivel de proyecto**
  - presenta las cuatro últimas funcionalidades expuestas a nivel de centro con la diferencia que solo intervienen los usuarios miembros de un proyecto.
  - permite asociar una competencia a un rol en un proyecto.

[1] Del acrónimo Generic Public License (Licencia publica general).

- registrar las asistencias mediante la fecha y hora, dirección IP y sección de trabajo, entiéndase esta como la jornada laboral mañana, tarde o noche.
- mostrar información básica de los Recursos Humanos de un proyecto, visualizar los usuarios, los roles asociados a ellos y el correo electrónico de los mismos.

1.2.2 Módulo de Creación de planes de formación basado en competencias laborales.

Este módulo presenta como desventaja que fue diseñado e implementado basándose en una versión ya desactualizada de la Suite de Gestión de Proyectos, esta deficiencia conlleva a que se dificulte la integración del mismo al módulo de Recursos Humanos, además de presentar duplicidad en funcionalidades existentes en este último. Este módulo tiene como característica fundamental la aplicación de un algoritmo que permite proponer planes de formación basados en competencias laborales. Al igual que el módulo de Recursos Humanos presenta funcionalidades a nivel de centro y de proyecto.

1.3 Metodologías, herramientas y lenguajes

Uno de los objetivos fundamentales a la hora de realizar una aplicación de *software* factible, eficaz y de calidad, es definir cuáles son las herramientas, lenguajes y metodologías que serán de mayor utilidad para la implementación. Las metodologías de desarrollo de *software* describen los pasos para lograr dicho objetivo, puesto que definen detalladamente qué se debe hacer y quién debe hacerlo. Precisan las tareas fundamentales a desarrollar a lo largo del ciclo de vida del proyecto, los artefactos que deben construirse, el orden en que se deben realizar y designan los responsables de cada tarea.

1.3.1 Metodología de desarrollo de software

Una metodología de desarrollo, en ingeniería de *software*, es un conjunto de herramientas, técnicas, procedimientos y soporte documental encaminada a estructurar, planificar y controlar el proceso de desarrollo de forma organizada y lógica que tienen como objetivo apoyar a los desarrolladores en la creación de un nuevo *software* (Kaisler, 2005).

Las metodologías de desarrollo se clasifican en dos clases: las metodologías tradicionales o robustas y las ágiles o ligeras (Letelier, et al., 2006). En la Tabla 2 se presenta una comparación entre los dos tipos de metodologías.

Metodologías tradicionales

Las metodologías tradicionales o prescriptivas definen un conjunto de actividades, acciones, tareas, fundamentos y productos de trabajo que se requieren para desarrollar *software* de alta calidad. El proceso conduce a un equipo de *software* a través de un conjunto de actividades del marco de trabajo que se organizan en un flujo de proceso, el cual puede ser lineal[2], incremental[3]o evolutivo[4]. La terminología y los detalles de cada modelo difieren, pero las actividades genéricas del marco de trabajo permanecen razonablemente consistentes.

Metodologías ágiles

Las metodologías ágiles combinan una filosofía y conjunto de directrices de desarrollo. La filosofía busca la satisfacción del cliente y la entrega temprana de *software* incremental, equipos de proyecto pequeños y con alta motivación, métodos informales, un mínimo de productos de trabajo de la ingeniería de *software*, y una simplicidad general de desarrollo. Las directrices de desarrollo resaltan la entrega sobre análisis y diseño (aunque estas actividades no se descartan) y la comunicación activa entre los desarrolladores y los clientes (Pressman, 2005).

En febrero del 2001 queda conformado durante una reunión en Utah, Estados Unidos, el Manifiesto Ágil. Este manifiesto recogía los principales principios y particularidades de este tipo de metodología, y afirmaba que en esta se valora:

- Al individuo y las interacciones del equipo de desarrollo, sobre el proceso y las herramientas.
- Desarrollar *software* que funcione, más que conseguir una buena documentación.
- La colaboración con el cliente, más que la negociación de un contrato.
- Responder a los cambios, más que seguir estrictamente un plan (Letelier, et al., 2006).

[2] Son aquellos modelos que desarrollan sus actividades de forma continua sin retrocesos a actividades previas y sin repetición de las ya ejecutadas.

[3] Modelos que responden a una situación donde los requisitos están bien definidos pero es necesario satisfacer al cliente de forma rápida con conjuntos limitados de funcionalidad en pequeñas porciones que aumentan gradualmente y se refinan y expanden en cada entrega.

[4] Modelos que responden a una situación donde los requisitos finales no están bien definidos, aunque sí el esquema general de necesidades del cliente pero es necesario satisfacer al cliente de forma rápida ante la presión del mercado, dedicándose inicialmente a satisfacer requisitos esenciales y luego trabajar sobre las extensiones de estos requisitos

Tabla 2 Comparación entre metodologías tradicionales y ágiles

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de *software*.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparadas para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo de desarrollo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas y normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (menos de 10) trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura de *software*.	La arquitectura de *software* es esencial y se expresa mediante modelos.

Entre las metodologías ágiles más destacadas hasta el momento se pueden nombrar:

- XP (*Extreme Programming*).
- Scrum.
- Crystal Clear.
- DSDM (*Dynamic Systems Developmemt Method*).
- FDD (*Feature Driven Development*).

- ASD (*Adaptive Software Development*).
- XBreed.
- Extreme Modeling.

Las metodologías ágiles están especialmente indicadas para proyectos con requisitos poco definidos o cambiantes. Estas se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación. Dividir el trabajo en módulos abordables minimiza los fallos y el coste. Es por ello por lo que se selecciona las metodologías ágiles para el desarrollo de esta investigación, además se tiene como factor influyente de que el sistema GESPRO se desarrolló utilizando este enfoque.

Scrum

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de *software* se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto. Estas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (Kniberg, 2007).

Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Es un proceso de desarrollo iterativo e incremental enfocado a la gestión de procesos de desarrollo de *software*. Se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del mismo. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, los requisitos son cambiantes o poco definidos. La innovación, la competitividad y la productividad son fundamentales (Albaladejo, 2009).

Metodología ágil, y como tal:

Es un modo de desarrollo de carácter adaptable más que predictivo.

Orientado a las personas más que a los procesos.

Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones (Palacio, 2006).

Propone las siguientes tres fases.

- **Fase de Planificación:**
  - Planeación: se define el equipo, herramientas, el sistema de desarrollo y se crea el productbacklog con la lista de requerimientos conocidos junto con sus prioridades y se estima el esfuerzo necesario para llevarlo a cabo.
  - Diseño Arquitectónico: se define la arquitectura del producto que permita implementar los requerimientos.
- **Fase de Desarrollo:**
  - Es la parte ágil, donde el sistema se desarrolla en Sprints[5].
- **Fase de Finalización:**
  - Incluye integración, testing y documentación. Indica la implementación de todos los requerimientos, quedando el productbacklog vacío (Mendes Calo, et al., 2010).

Pila de Productos

La pila de producto es una lista priorizada de funciones que el producto debe tener. La prioridad de las funciones siempre estará sujeta a cambios, debido a que hay elementos que dependen de otros y continuamente aparecen nuevos elementos que antes se desconocían (Schwaber, et al., 2013).

Pila de Sprint

Para realizar un Sprint se cogen una serie de elementos de la Pila de Productos (los que se prevén posibles de realizar en el tiempo estimado) y se construye con ellos la Pila de Sprint. Al crearla habrá que eliminar los elementos seleccionados de la Pila de Producto, y desglosarlos en tareas concretas, a las que se les puede medir el esfuerzo que llevará realizarlas (Schwaber, et al., 2013).

[5] Un sprint (o iteración) es la unidad básica del desarrollo en Scrum, el mismo se denomina a cada iteración de desarrollo y se recomienda realizarlas con duraciones de 30 días.

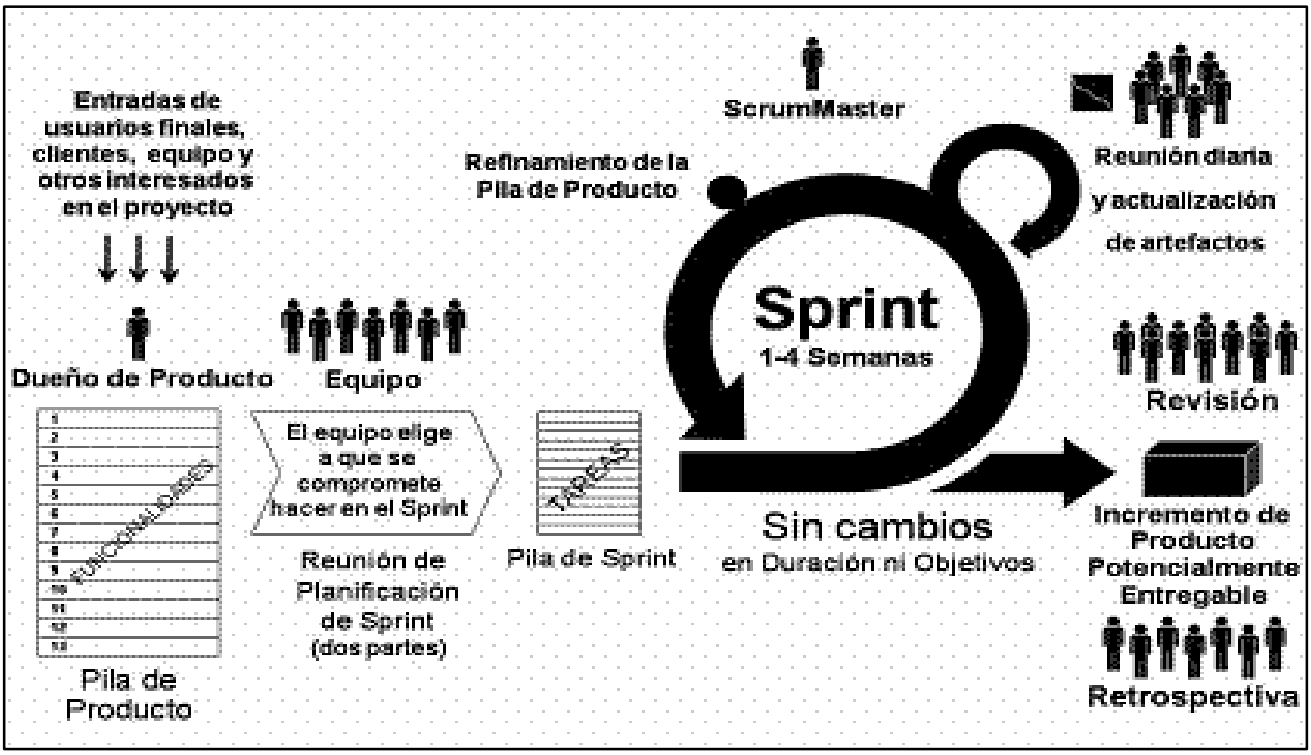


Figura 1 Roles, artefactos y eventos principales de SCRUM (Pete Deemer, 2009)

Luego del análisis de las anteriores características se decide que Scrum sea la que dirija el proceso de desarrollo del componente en cuestión puesto que es una metodología de desarrollo muy simple y por tal motivo está dirigida a proyectos con requisitos inestables y que requieren rapidez y flexibilidad, como es el caso. La misma requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Importante mencionar que el componente a rediseñar fue desarrollado usando esta metodología de desarrollo.

1.3.2 Lenguaje para el modelado

En el proceso de desarrollo de *software* se hace de carácter fundamental e imprescindible el modelado del mismo, este le proporciona al desarrollador un conjunto de herramientas, artefactos y notaciones que le garantizan una “idea visual” del sistema en construcción, lográndose una lógica de este último.

UML

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y

controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar (Object Management Group, 2014).

UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos (Rumbaugh, et al., 2007).

UML no es un lenguaje de programación, es un lenguaje de modelado universal, discreto y de propósito general (Rumbaugh, et al., 2000).

1.3.3 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser ejecutados por máquinas computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (Wilson, 1993).

Ruby es un lenguaje con un balance cuidado. Su creador, Yukihiro “Matz” Matsumoto, mezcló partes de sus lenguajes favoritos (Perl, Smalltalk, Eiffel, Ada, y Lisp) para formar un nuevo lenguaje que incorporara tanto la programación funcional como la programación imperativa. Orientado a objetos absolutamente puro escrito en C y diseñado con Perl.

Características generales del lenguaje:

1. Orientado a objetos.
2. Cuatro niveles de ámbito de variable: global, clase, instancia y local.
3. Manejo de excepciones.
4. Iteradores y clausuras o closures (pasando bloques de código).
5. Expresiones regulares nativas similares a las de Perl a nivel del lenguaje.
6. Posibilidad de redefinir los operadores (sobrecarga de operadores).
7. Recolección de basura automática.

8. Altamente portable.
9. Hilos de ejecución simultáneos en todas las plataformas usando greenthreads.
10. Carga dinámica de DLL/bibliotecas compartidas en la mayoría de las plataformas.
11. Introspección, reflexión y metaprogramación.
12. Amplia librería estándar.
13. Soporta inyección de dependencias.
14. Soporta alteración de objetos en tiempo de ejecución.
15. Continuaciones y generadores (Matsumoto, 2001).

Para llevar a cabo el proceso de rediseño se escoge este lenguaje por su dinamismo y fortaleza, además de brindar la posibilidad de ser código abierto enfocado en la simplicidad y productividad y en particular la plataforma XEDRO GESPRO fue implementada en dicho lenguaje por tales motivos todo nuevo componente que se le desee agregar deberá estar desarrollado en el mismo lenguaje.

1.3.4 Herramientas y tecnologías

Para el desarrollo de esta aplicación, se realizó un estudio de las principales herramientas y tecnologías que se seleccionaron para la implementación del mismo, teniendo en cuenta las especificaciones del cliente y siguiendo los estándares empleados en el proyecto GESPRO.

1.3.4.1 Herramientas para el diseño

Las herramientas **CASE** (***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*, Ingeniería de Software Asistida por Computadora) constituyen aplicaciones informáticas dirigidas a aumentar la productividad en el desarrollo de *software*, reduciendo el costo del mismo en términos de tiempo y de dinero. Estas herramientas pueden soportar todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores (Muller, et al., 2011).

Rational Rose

Rational Rose es una poderosa herramienta de modelado visual para apoyar el análisis y diseño de sistemas del *software* orientados a objeto. Usando el modelado, se pueden atrapar desperfectos del diseño a tiempo, mientras no son costosos de reparar. Describe con todo lujo de detalles lo que el sistema incluirá y cómo trabajará, así que los desarrolladores pueden usar el modelo como una heliografía para el sistema en construcción. Incluye todos los diagramas UML (Rational Software, 2009).

El *software* permite acelerar el desarrollo de muchas aplicaciones con código generado a partir de modelos visuales mediante el lenguaje UML. Ofrece una herramienta y un lenguaje de modelado común para simplificar el entorno de trabajo y permitir una creación más rápida de *software* de calidad (Rational Software, 2009).

Desarrollo más rápido de las aplicaciones: contiene un entorno de modelado visual que permite agilizar el desarrollo de aplicaciones.

Integración del diseño de aplicaciones con el desarrollo: unifica el equipo del proyecto proporcionando una ejecución y una notación de modelos UML comunes (Rational Software, 2009).

Visual Paradigm

Visual Paradigm es una herramienta CASE para el modelado de lenguaje UML que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (Visual Paradigm International, 2013).

Visual Paradigm también ofrece:

Navegación intuitiva entre la escritura del código y su visualización.

Potente generador de informes en formato PDF/HTML.

Documentación automática Ad-hoc.

Ambiente visualmente superior de modelado.

Sofisticado diagramador automáticamente de *layout*.

Sincronización de código fuente en tiempo real (Targetware, 2013).

Elección de la herramienta de modelado

Una vez analizadas estas dos potentes herramientas CASE para el modelado de lenguaje UML, se escoge Visual Paradigm porque ofrece las características necesarias para realizar el diseño de la solución del presente trabajo. Entre estas características se pueden citar el sofisticado diagramador automático de plantillas y el ambiente visualmente superior de modelado que serán de gran ayuda para el trabajo con esta herramienta.

Otro aspecto significante tenido en cuenta para esta elección estuvo dado por su fácil manejo y uso, además de la experiencia adquirida por trabajos anteriores realizados con dicha herramienta.

1.3.4.2 Framework Ruby on Rails (*RoR*)

En general, con el término *Framework*, nos estamos refiriendo a una estructura *software* compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, se puede considerar como una aplicación genérica, incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta (Rodriguez Gutíerrez, 2007).

Los objetivos principales que persigue un *Framework* son: **acelerar** el proceso de desarrollo, **reutilizar** código ya existente y **promover** buenas prácticas de desarrollo como el uso de patrones. Un *Framework* web, por tanto, podemos definirlo como un conjunto de componentes (por ejemplo clases en Java, descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas web (Rodriguez Gutíerrez, 2007).

RoR es un *Framework* de desarrollo de aplicaciones web escrito en el lenguaje Ruby. Está diseñado para desarrollar aplicaciones web de manera más fácil al hacer suposiciones acerca de lo que necesita cada desarrollador para empezar. Permite escribir menos código durante el cumplimiento de más de muchos otros lenguajes y Frameworks. Experimentados desarrolladores de Rails también informan de que el desarrollo de aplicaciones web se produzca de forma más divertida (Hansson).

1.3.4.3 Entorno de desarrollo integrado

Se define como entorno de desarrollo integrado (IDE), por sus siglas en inglés: Integrated Development Environment, al programa compuesto por un conjunto de herramientas para un programador, es decir, consiste en un editor de código, un compilador, un depurador y en algunos casos un constructor de interfaz gráfica (GUI). Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios de ellos (Salavert, et al., 2000).

Un IDE debe tener las siguientes características:

- Multiplataforma.
- Soporte para diversos lenguajes de programación.
- Integración con Sistemas de Control de Versiones.
- Reconocimiento de Sintaxis.

- Extensiones y Componentes para el IDE.
- Integración con *Framework*s populares.
- Depurador.
- Importar y Exportar proyectos.
- Múltiples idiomas.
- Manual de Usuarios y Ayuda.

NetBeans

NetBeans IDE es un entorno de desarrollo visual de código abierto, libre y gratuito sin restricciones de uso. Es utilizado para aplicaciones programadas mediante Java, uno de los lenguajes de programación más poderosos del momento, aunque puede ser utilizado para programar en otros lenguajes, entre ellos *C* / *C* + +, *PHP*, JavaScript y Ruby. Dentro de sus características podemos encontrar también, que es multiplataforma y con él es posible desarrollar desde aplicaciones para la Web, para dispositivos portátiles, como móviles o Pocket PC, hasta potentes aplicaciones de escritorio (Korgent Inc., 2008).

Es seleccionado como entorno de desarrollo porque permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de *software* llamados módulos. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos, además contempla el lenguaje a utilizar para la implementación del rediseño del módulo de Recursos Humanos permitiendo el completamiento de código y la integración del mismo con el entorno de desarrollo. Otro factor influyente en esta selección es la facilidad de uso para la programación web y orientada a objetos. Como característica adicional se puede mencionar que presenta una excelente integración con el *Subversion*[6] y se ejecuta en varias plataformas, incluyendo *Windows*, *Linux*, *Mac OS X* y *Solaris*.

1.3.4.4 Sistema Gestor de Base de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es un sistema de *software* cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones, además de permitir a los usuarios definir, crear y mantener la base de datos y proporcionar un acceso controlado a la misma (Fidel Gil, 2005).

[6] Sistema de control de versiones de código abierto fundada en 2000 por CollabNet, Inc.

PostgreSQL es un potente sistema de base de datos, de código abierto objeto-relacional. Desarrollado bajo la licencia de *software* otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*).Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad, integridad de datos y corrección. Funciona en todos los principales sistemas operativos, incluyendo *Linux*, *UNIX* (*AIX*, *BSD*, *HP-UX*, *SGI IRIX*, *Mac OS X*, *Solaris*, *Tru64*) y *Windows*. Dispone de interfaces de programación nativas para C / C + +, Java, Net, Perl, Python, Ruby, Tcl, ODBC, entre otros, y una documentación excepcional. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para garantizar la estabilidad del sistema (PostgresSQL, 2011). Incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores y reglas e integridad transaccional. A pesar de esto, *PostgreSQL* no es un sistema de gestión de bases de datos puramente orientado a objetos (High Tech Tools Providers del Perú, 2012).

PgAdmin3 es una aplicación de diseño y manejo de bases de datos para su uso con *PostgreSQL*. La aplicación está escrita en *C++* y funciona sobre casi todas las plataformas. La interfaz gráfica es compatible con todas las características de *PostgreSQL* y facilita la administración. La conexión del servidor se puede realizar mediante *TCP/IP* o *Unix Domain Sockets* y puede ser cifrado mediante SSL por seguridad. No se requieren controladores adicionales para comunicarse con la base de datos del servidor e incorpora funcionalidades para realizar consultas, examinar su ejecución y trabajar con los datos (PgAdmin III, 2012).

Conclusiones parciales

La definición de los conceptos de reingeniería, rediseño y mejora continua de procesos permitió establecer el impacto sobre el cual recae la solución del problema en el módulo de Recursos Humanos.

El análisis de los principales conceptos relacionados a la investigación, brindó un mayor conocimiento sobre el campo de acción en el que se desarrollará la solución.

La definición de la metodología Scrum, para el desarrollo del módulo, posibilitó ajustar las funcionalidades basándose en las necesidades del cliente y permitió la visualización el proyecto día a día.

Las herramientas y tecnologías de desarrollo de *software* permitieron sentar las bases teóricas y tecnológicas para dar solución al problema planteado en la investigación.

Capítulo 2: Análisis y diseño de la solución propuesta

Introducción

En el presente capítulo se exponen las principales características del rediseño del módulo de Recursos Humanos. Se realiza la captura de los requisitos funcionales y no funcionales. Se describen los patrones arquitectónicos y de diseño a aplicar y se ilustran diagramas tanto de clases del diseño, de componentes y de diseño de tablas de la base de datos.

2.1 Esencia del análisis y diseño

La esencia del **análisis** y **el diseño orientado a objetos** consiste en situar el dominio de un problema y su solución lógica dentro de la perspectiva de los objetos (cosas, conceptos o entidades), como se advierte en la siguiente figura.

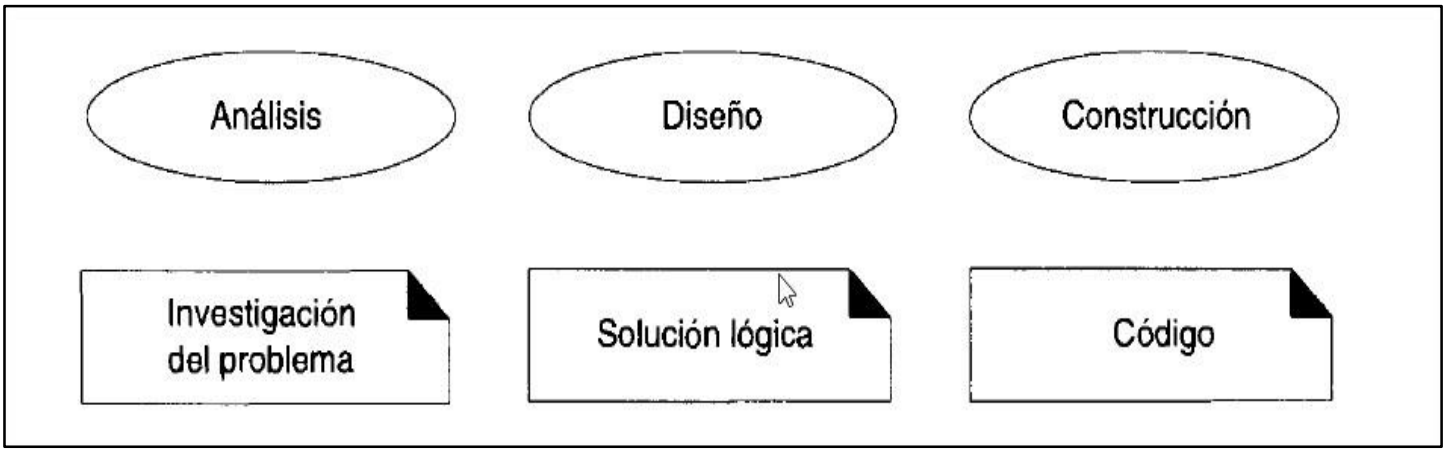


Figura 2 Significado de las actividades de desarrollo.

Durante el **análisis orientado a objetos** *se* procura ante todo identificar y describir los objetos o conceptos dentro del dominio del problema.

Durante el **diseño orientado a objetos,** se procura definir los objetos lógicos del *software* que finalmente serán implementados en un lenguaje de programación orientado a objetos.

Finalmente, durante la **construcción** o **programación orientada a objetos,** se implementan los componentes del diseño. Esta última será indagada en el próximo capítulo.

2.2 Propuesta de solución

Al definir la gestión de Recurso Humanos como un conjunto de actividades, entre las cuales la formación del personal vinculado a una organización desempeña un papel protagónico, se propone como solución de la presente investigación rediseñar e implementar el módulo para la gestión de Recurso Humanos de la plataforma XEDRO GESPRO, de manera tal, que permita gestionar las actividades de formación según la necesidad de mejorar el rendimiento presente o futuro, aumentando así la capacidad de conocimiento, de la fuerza de trabajo.

El nuevo módulo deberá ser capaz de mostrar las brechas entre el nivel alcanzado por el usuario en una competencia (nivel real) y el nivel óptimo para realizar la misma (nivel ideal). Tendrá la capacidad de gestionar actividades de formación, atendiendo a las necesidades de cada proyecto, además de proponer, según las brechas entre los niveles, un plan de formación para disminuir tales brechas. Ver en la siguiente figura el mapa conceptual de la solución propuesta.

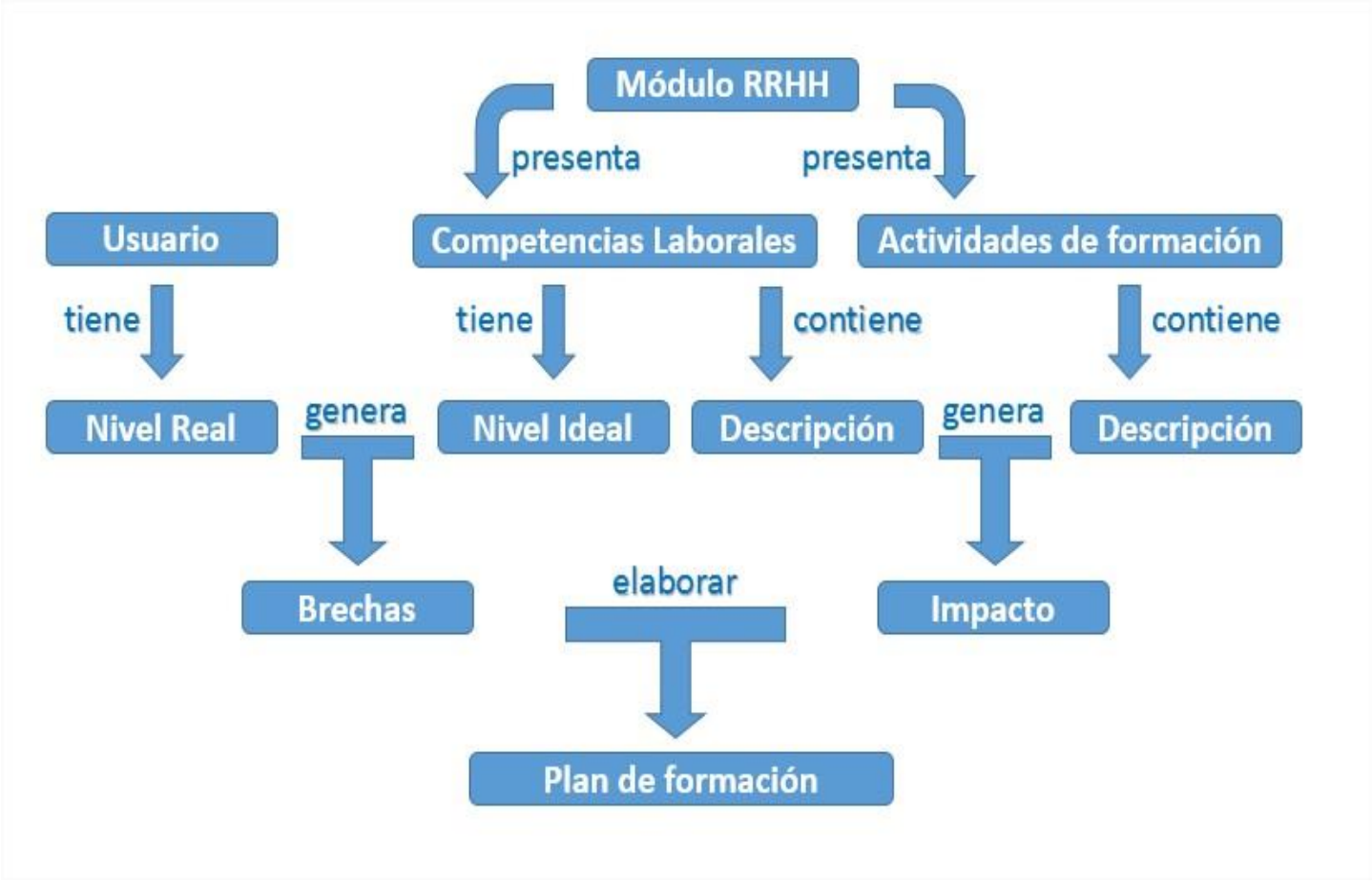


Figura 3 Mapa conceptual de la propuesta de solución.

2.3 Especificación de requisitos

La escritura de los requisitos se refiere a la tarea de reunir una descripción del producto desde el punto de vista de negocio. Los requisitos obtenidos fueron especificados de manera correcta en la plantilla *0113_Especificación De Requisitos De Software* con la cual quedan descritos de manera correcta los requisitos identificados para el rediseño del módulo Recursos Humanos del sistema XEDRO GESPRO (para obtener mayor detalle dirigirse al documento adjunto a la investigación. Ver Anexo 1).

2.3.1 Requisitos funcionales

Los requisitos funcionales describen lo que el sistema o el *software* debe de hacer. La funcionalidad es la capacidad útil proporcionada por uno o más componentes de un sistema. En algunos casos, los requisitos funcionales también pueden declarar explícitamente lo que el sistema no debe hacer (Socas Alvez, et al., 2007).

El sistema a rediseñar cuenta con una serie de requerimientos funcionales, los cuales serán modificados algunos de ellos y otros serán agregados según a la descripción del negocio.

Requisitos:

RF 1 Mostrar la diferencia entre el nivel real y el nivel deseado (Brecha) de cada competencia por individuo.

RF 2 Gestionar actividades de formación.

RF 2.1 Crear actividad de formación.

RF 2.2 Modificar actividad de formación.

RF 2.3 Eliminar la actividad de formación.

RF 2.4 Mostrar información de una actividad de formación.

RF 3 Mostrar el impacto de las actividades de formación sobre las competencias laborales.

RF 4 Mostrar el costo de realización y el costo de matrícula de las actividades de formación.

RF 5 Gestionar planes de formación.

RF 5.1 Crear plan de formación.

RF 5.2 Eliminar plan de formación.

RF 5.3 Mostrar plan de formación.

RF 6 Registrar las asistencias de los usuarios.

RF 7 Mostrar el resumen de asistencias por concepto de usuario.

RF 8 Mostrar las evaluaciones de las competencias laborales por usuario.

RF 9 Mostrar la evaluación de las competencias de los usuarios pertenecientes a un proyecto.

2.3.2 Requisitos no funcionales

Tabla 3 Requisitos no funcionales.

Hardware	
RNF1	Servidor Aplicaciones Web: 2GB RAM, 250 GB disco duro.
RNF2	Servidor Base de Datos: 2GB RAM, 250 GB disco duro.
RNF3	PC Cliente: Prestaciones de hardware que permitan la instalación de cualquier Sistema Operativo, conexión de red y un navegador para acceder a la plataforma.
Software	
RNF4	PC Cliente con navegador Mozilla Firefox v 13.0 o superior.
RNF5	PC servidor de base de datos: El servidor debe contar con sistema operativo GNU/Linux Ubuntu 12.04 LTS y el SGDB PostgreSQL 9.1.
RNF6	Sistema de Control de Versiones: *Subversion* 2.6.
RNF7	Como servidor de aplicación Apache 2.
Restricciones en el diseño y la implementación	
RNF8	IDE de desarrollo: NetBeans 6.9.1.
RNF9	Paradigma de la arquitectura MVC.

RNF10	Lenguaje de programación: Ruby.
RNF11	Navegador Web: Mozilla Firefox v 6.0 o superior.
Eficiencia	
RNF12	El sistema debe soportar un tiempo de respuesta menor o igual a 5 segundos.
RNF13	El sistema debe soportar una conexión simultánea de más de 3000 usuarios.
Requisitos para la documentación de usuarios en línea y ayuda del sistema	
RNF14	Manual de usuario: el sistema debe tener un manual de usuario, permitiendo con ello un correcto uso de sus funcionalidades y brindarle al usuario una mayor experiencia del trabajo con el mismo.
RNF15	Documentación actualizada del grupo de desarrollo: se precisa que la documentación del sistema esté actualizada en todos los aspectos, fases de trabajo y ciclos de desarrollo del mismo, permitiendo con ello un respaldo tanto ingenieril como legal del desarrollo de dicho sistema.
RNF16	El sistema brinda como apoyo una ayuda contextual en la cual se refleja detalladamente la explicación de cada una de las pantallas con sus respectivas funcionalidades.
Requisitos de apariencia o interfaz externa	
RNF17	La interfaz es sencilla con colores suaves a la vista y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo.
RNF18	Debe utilizar como idioma principal el español, aunque debe existir traducción al inglés.
RNF19	Los botones expresarán su función ya sea mediante su texto o la imagen que lo acompañe.
Interfaces de hardware	
RNF20	La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo de conexión TCP/IP.

RNF21	La comunicación entre el cliente y el servidor de aplicaciones se lleva a través del protocolo HTTPS.
Requisitos Legales, de Derecho de Autor y otros	
RNF22	El sistema debe ser sometido a un análisis legal por parte de los abogados y personal autorizado con vistas a declarar su autenticidad y evitar restricciones legales para su uso y comercialización; así mismo se debe proceder a una evaluación y certificación por parte del cliente del producto.
Seguridad	
RNF23	El sistema debe estar disponible las 24 horas los 365 días del año, para garantizar la ejecución de las tareas en el momento requerido.
RNF24	Permite el acceso al sistema a través de una cuenta de usuario única que puede ser local o por autenticación con el LDAP.
RNF25	El acceso por las cuentas de administración a los servidores se realiza a través de una contraseña generada con algoritmo MD5. El listado de las claves de cada servidor se almacena en un archivo comprimido cifrado.
RNF26	Los usuarios creados en el sistema XEDRO GESPRO tienen acceso (o permiso) limitado, podrán acceder a un conjunto de funcionalidades según el rol o perfil donde fue clasificado el usuario, que a su vez este rol fue clasificado en un nivel para la toma de decisiones determinado.
Usabilidad	
RNF27	Facilidad de uso por parte de los usuarios: el sistema debe presentar una interfaz amigable que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Debe, además, ser una interfaz de manejo cómoda que posibilite a los usuarios sin experiencia una rápida adaptación.
RNF28	Especificación de la terminología utilizada: el sistema debe adaptarse al lenguaje y términos utilizados por los usuarios en la rama abordada con vista a una mayor comprensión por parte del cliente de la herramienta de trabajo.

RNF29	Potencialidades de capacitación orientadas a interfaces intuitivas, lo que enaltece la posibilidad de que el usuario aprenda mediante el uso y explotación de la herramienta.
RNF30	El sistema debe presentar una serie de menús tanto laterales como en barra de iconos flotantes que permitan el acceso rápido a la información por parte de los usuarios, lo que aprovecha las potencialidades de estas estructuras.
Portabilidad	
RNF 31	El sistema deberá permitir su uso en diferentes sistemas operativos, como Linux y Windows, con navegador Mozilla Firefox v 13.0 o superior.

2.4 Diseño del módulo

El diseño es una representación de la ingeniería en el desarrollo de aplicaciones. Es el lugar en donde se fomentará la calidad de la ingeniería, en el cual se proporcionan las representaciones del *software* que se pueden evaluar en cuanto a calidad. Es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema finalizado y sirve como fundamento para todos los pasos siguientes del soporte y de la ingeniería del *software*.

Para lograr un sistema *software* de buena calidad es necesario lograr un diseño de excelencia. A continuación se exponen los fundamentos del rediseño del módulo de Recursos Humanos para el sistema GESPRO, utilizando como punto de partida el diseño estructural del módulo actual y el nuevo diseño propuesto, la arquitectura del mismo seguido de los patrones de diseño empleados y los diferentes artefactos generados en el proceso de diseño e implementación.

En la siguiente figura se muestra el diseño de la estructura del módulo actualmente en función, el cual será analizado y reestructurado para lograr el objetivo del presente trabajo.

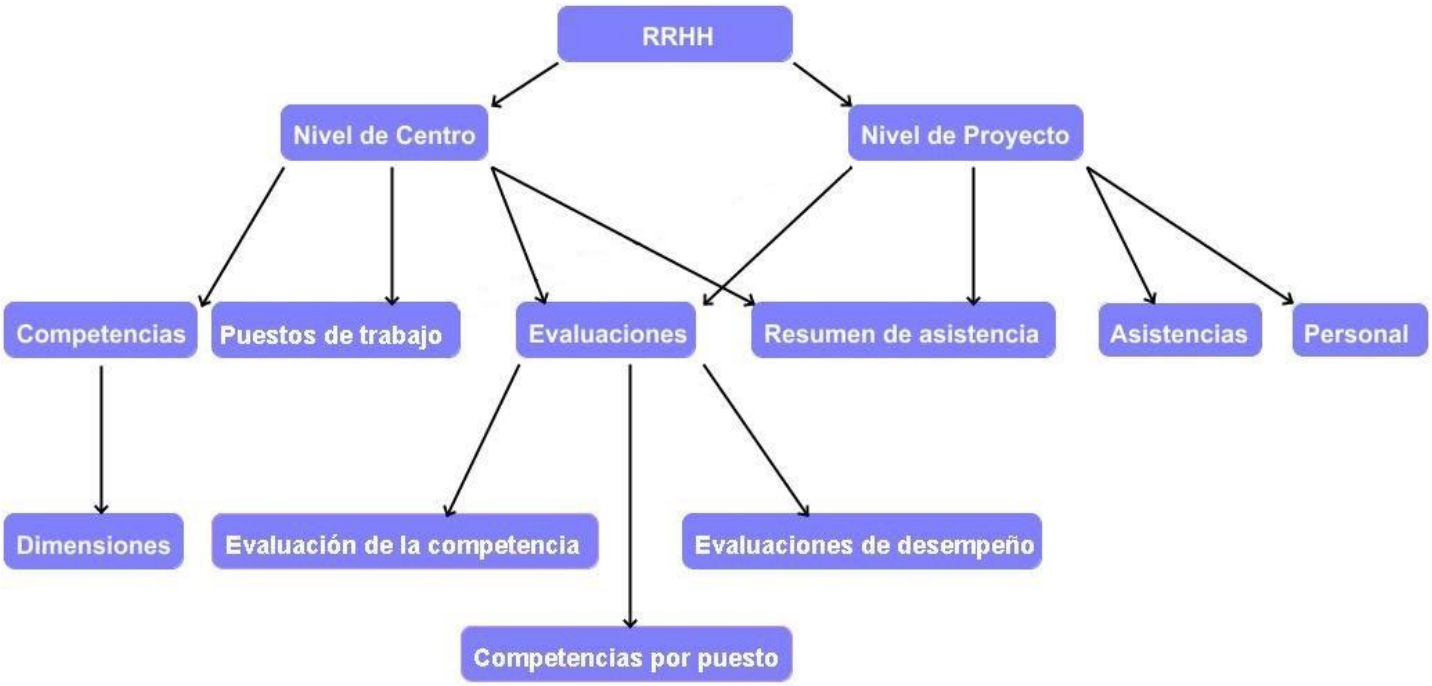


Figura 4 Estructura actual del módulo de Recursos Humanos de la herramienta GESPRO 13.05.

Después de analizada la estructura diseñada anteriormente se decide conformar una nueva estructura que logre dar cumplimiento al objetivo general del vigente trabajo. Ver Figura 5.

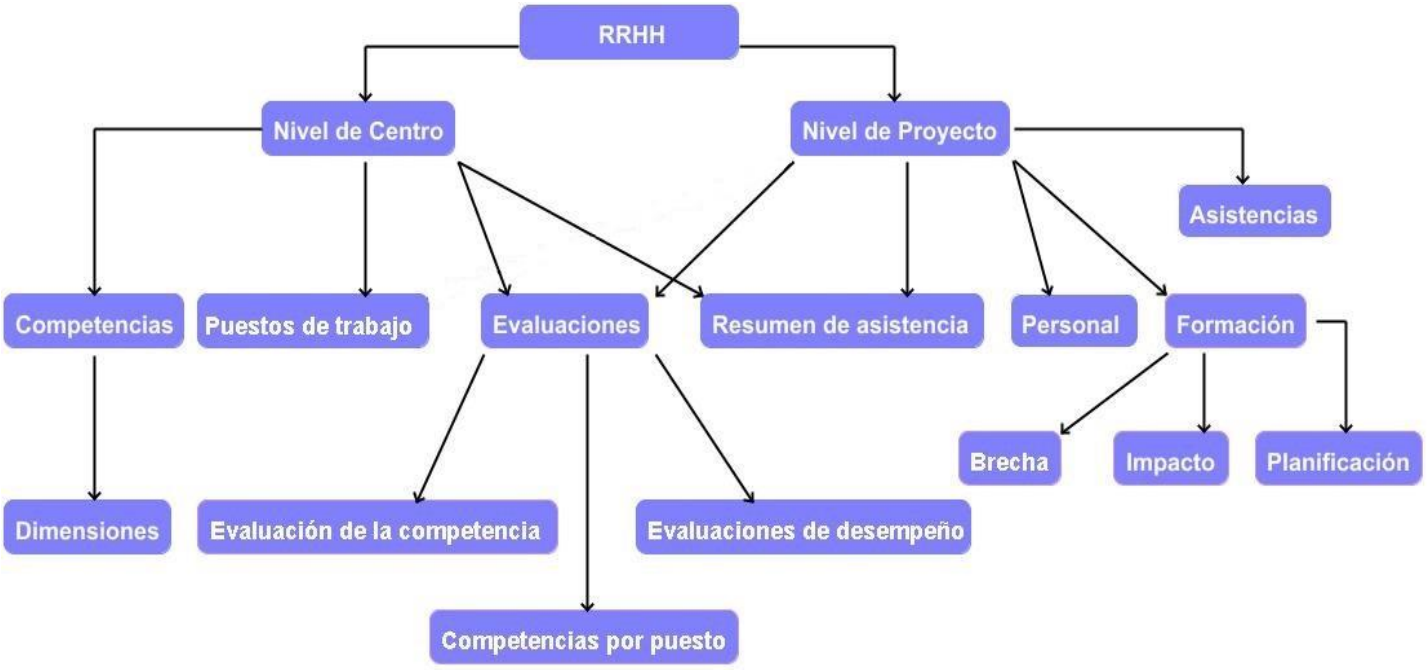


Figura 5 Estructura propuesta para el futuro módulo de Recursos Humanos para el sistema GESPRO.

2.4.1 Patrón de arquitectura

Los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de *software*. Provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos (Buschmann, et al., 1996).

Modelo Vista Controlador

El Modelo Vista Controlador (MVC) es un patrón de diseño de arquitectura que se usa principalmente en aplicaciones que procesan gran cantidad de datos y transacciones complejas, donde se requiere una mejor separación de conceptos para que el desarrollo esté estructurado de manera eficiente, facilitando así la programación en diferentes capas y de manera paralela e independiente.

Modelo (*Model*): Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). *RoR* define en un directorio llamado "*models*" todas las clases consideradas como modelos, que se utilizan para la comunicación con la base de datos.

Vista (*Views*): Es responsable de generar una interfaz de usuario, normalmente, basada en los datos del modelo. En el caso de una aplicación Web, la Vista es una página HTML con contenido dinámico sobre el cual el usuario puede realizar distintas operaciones. *RoR* define un conjunto de vistas en un directorio llamado "*views*" para cada funcionalidad básica, en un sub-directorio con el mismo nombre del controlador al que responden.

Controlador (*Controller*): Se encarga de manejar y responder las solicitudes del usuario interactuando de esta manera con el modelo para procesar la información necesaria, o sea, organiza la aplicación recibiendo eventos del exterior. Los controladores administran las acciones de *RoR*, y se encuentran en un directorio llamado "*controllers*".

El uso del patrón MVC servirá de apoyo para estructurar los componentes del módulo de manera tal que se obtenga una clara separación entre interfaz, lógica de negocio y de presentación, lo cual brindará una mayor sencillez para crear distintas representaciones de los datos, dará soporte a la reutilización de los componentes y brindará simplicidad para un mantenimiento futuro.

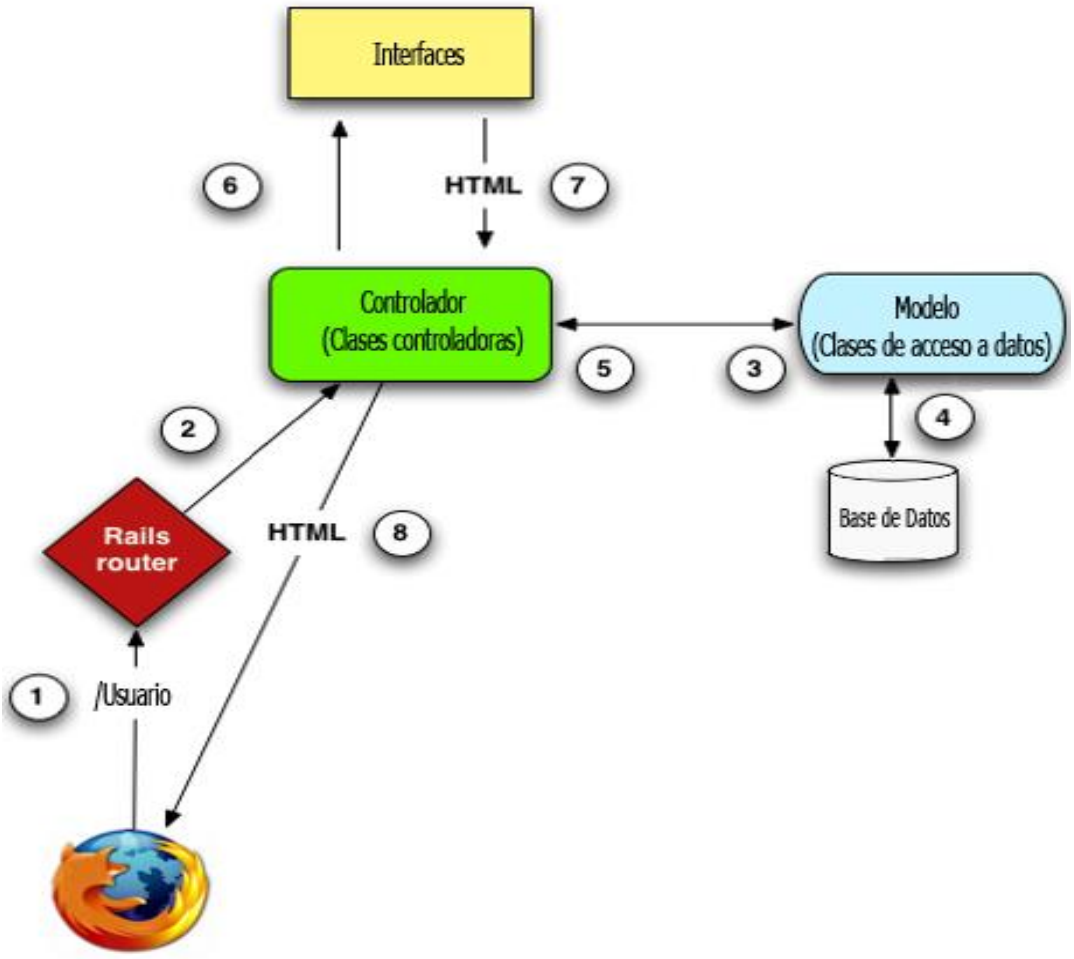


Figura 6 Arquitectura del *Framework RoR* basado en el patrón MVC.

2.4.2 Patrón de diseño

Los patrones de diseño hacen que sea más fácil de reutilizar los diseños y arquitecturas expresando técnicas probadas, lo cual facilita el trabajo a los desarrolladores de los nuevos sistemas. Para la realización de este trabajo serán de gran ayuda para elegir las alternativas de diseño que hacen que un sistema sea reutilizable y así de esta forma evitar las alternativas que ponen en peligro la reutilización (Gamma, et al., 1994).

Los patrones de diseño pueden incluso mejorar la documentación y el mantenimiento de los sistemas existentes mediante el suministro de una especificación explícita de la clase y la interacción de objetos y su intención subyacente. En pocas palabras, los patrones de diseño ayudan a un diseñador obtener un diseño de "correcta" más rápido.

Patrones GRASP

Los patrones GRASP acrónimo que significa *General Responsibility Asignment Software Patterns* (patrones generales de *software* para asignar responsabilidades) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

El **patrón Creador** guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

El uso de este patrón se refleja en el módulo de Recursos Humanos en las clases modelos, ejemplos de ellas se tienen: *GesproTrainingActivity, GesproCompetence, GesproCompetenceUser* y *GesproDimension*. Las mismas son encargadas de crear objetos de tipo modelo que son utilizados para realizar peticiones a la base de datos.

El **Bajo Acoplamiento** estimula asignar una responsabilidad, de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad.

Alta Cohesión es un principio que se debe tener presente en todas las decisiones de diseño: es la meta principal que ha de buscarse en todo momento. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas.

Los patrones Bajo Acoplamiento y Alta Cohesión antes descritos son utilizados para diseñar e implementar las clases del módulo de Recursos Humanos, brindando de este modo una mayor asignación de responsabilidades a cada una de ellas, evitando que las mismas se sobrecarguen de funcionalidades y solo estén relacionadas con aquellas que sean necesarias para llevar a cabo sus funciones. El uso del mismo se observa en todas las clases del módulo.

Controlador este patrón define que clase administra los eventos del sistema. Un controlador es un objeto que no pertenece a la interfaz de usuario, responsable de recibir o manejar un evento del sistema. Su uso se evidencia en las clases controladoras, pues estas tienen la responsabilidad de recibir o manejar los eventos del

sistema. En la aplicación se dividen los eventos del sistema en varias clases controladoras para lograr disminuir el acoplamiento y aumentar la cohesión.

Este patrón es utilizado en el módulo de Recursos Humanos en las clases controladoras, ejemplos de ellas se tienen: *GesproCompetenceController*, *GesproDimensionController* y *GesproTrainingController*. Las mismas son las responsables de controlar los eventos del módulo en cuanto a atender peticiones generadas por los usuarios.

2.4.3 Patrón de acceso a datos

El patrón de Acceso a Datos es un componente de *software* que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos (Mesa, 2012). Entre sus características está que el objeto contenga los datos presentes en una fila de la tabla que represente. Además, el objeto debe encapsular la lógica necesaria para interactuar con la base de datos. De esta forma, el acceso a datos se realiza de una forma más sencilla y uniforme.

Active Record (AR) sigue el estándar de Mapeo Objeto-Relacional (ORM) y se diferencia de los demás porque minimiza la cantidad de configuraciones mediante el uso de un conjunto de convenciones (Ramos, 2010). Cada clase AR representa una tabla de la base de datos cuyos atributos son representados como las propiedades de la clase y una instancia AR representa una fila en esa tabla.

Las operaciones CRUD[7] comunes son implementadas como métodos de la clase. Como resultado, podemos acceder a nuestros datos de una manera más orientada a objetos (Yiiframework, 2010).

Este patrón se encuentra implementado en el módulo de Recursos Humanos en las clases modelos para acceder a los datos de la base de datos. Un ejemplo es la clase *GesproTrainingActivity*, que se encuentra en el archivo *gespro_training_activity.rb* en la carpeta *models*, y la misma hereda de *ActiveRecord::Base*. Los cambios que sean necesarios se realizan en la base de datos mediante los objetos de *ActiveRecord* como se puede ver en la figura 7**.**

[7] CRUD: acrónimo de Crear, Obtener, Actualizar y Borrar (del original en inglés: *Create*, *Read*, *Update* and *Delete*), es usado para referirse a las funciones básicas en bases de datos o la capa de persistencia en un *software*.

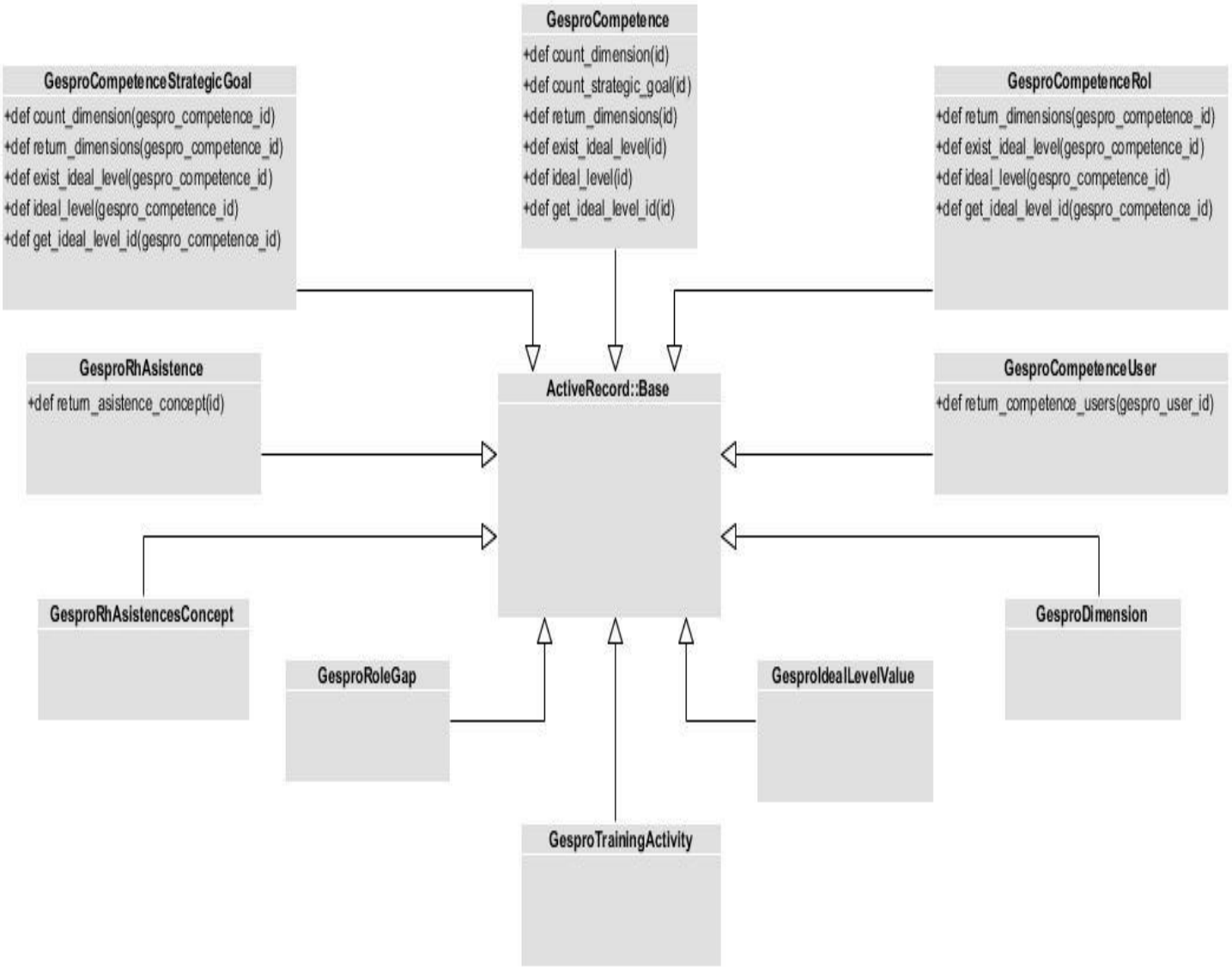


Figura 7 Representación del uso del patrón Active Record.

2.4.4 Diagramas de clases del diseño

Uno de los aspectos fundamentales y el cual es de relativa importancia para la gestión de Recurso Humanos de cualquier empresa es la gestión las actividades de formación dirigidas a los miembros de un proyecto con el objetivo de capacitar el personal para disminuir las brechas que se generan entre los niveles reales de los usuarios y los niveles ideales de las competencias laborales. En las figuras 8, 9 y 10, diagramas de diseño de clases, se puede observar cómo quedaría diseñada la estructura para gestionar estas actividades de formación.

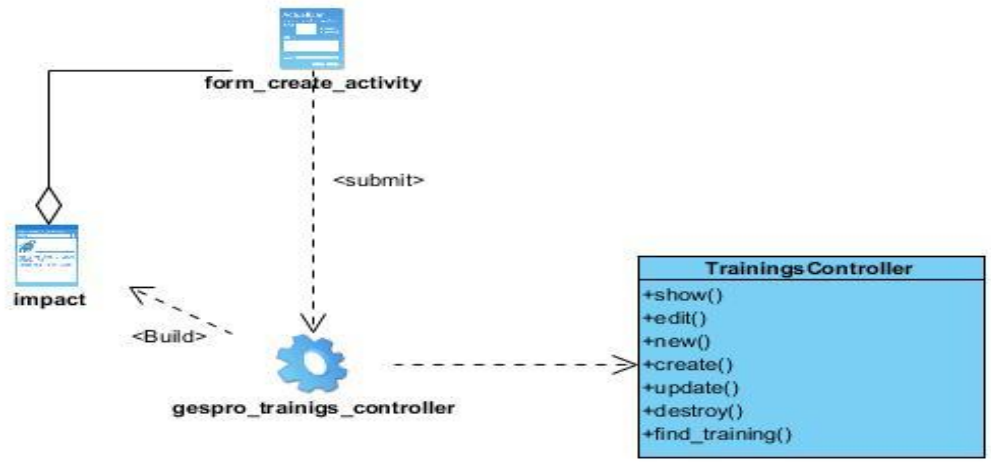


Figura 8 Crear actividad de formación.

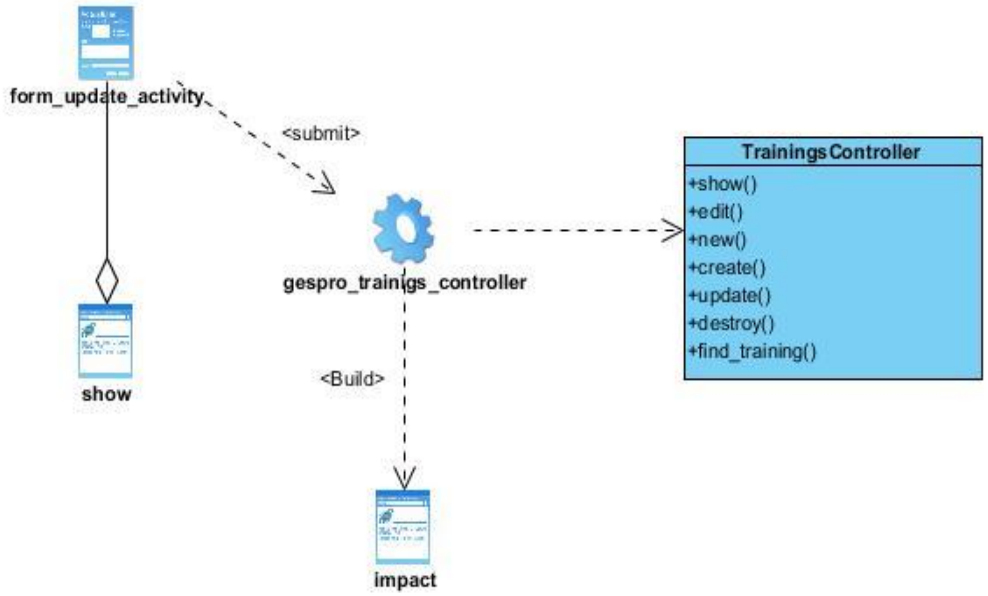


Figura 9 Modificar actividad de formación.

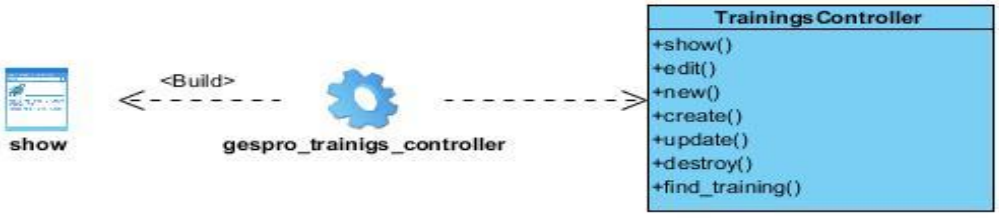


Figura 10 Mostrar actividad de formación.

Para alcanzar la meta propuesta por el objetivo del presente trabajo, permitir la propuesta de planes de formación basados en competencias laborales, es necesario agregar nuevas tablas a la base de datos. En la figura 11 se especifican las tablas existentes y por último en la figura 12 las nuevas a agregar.

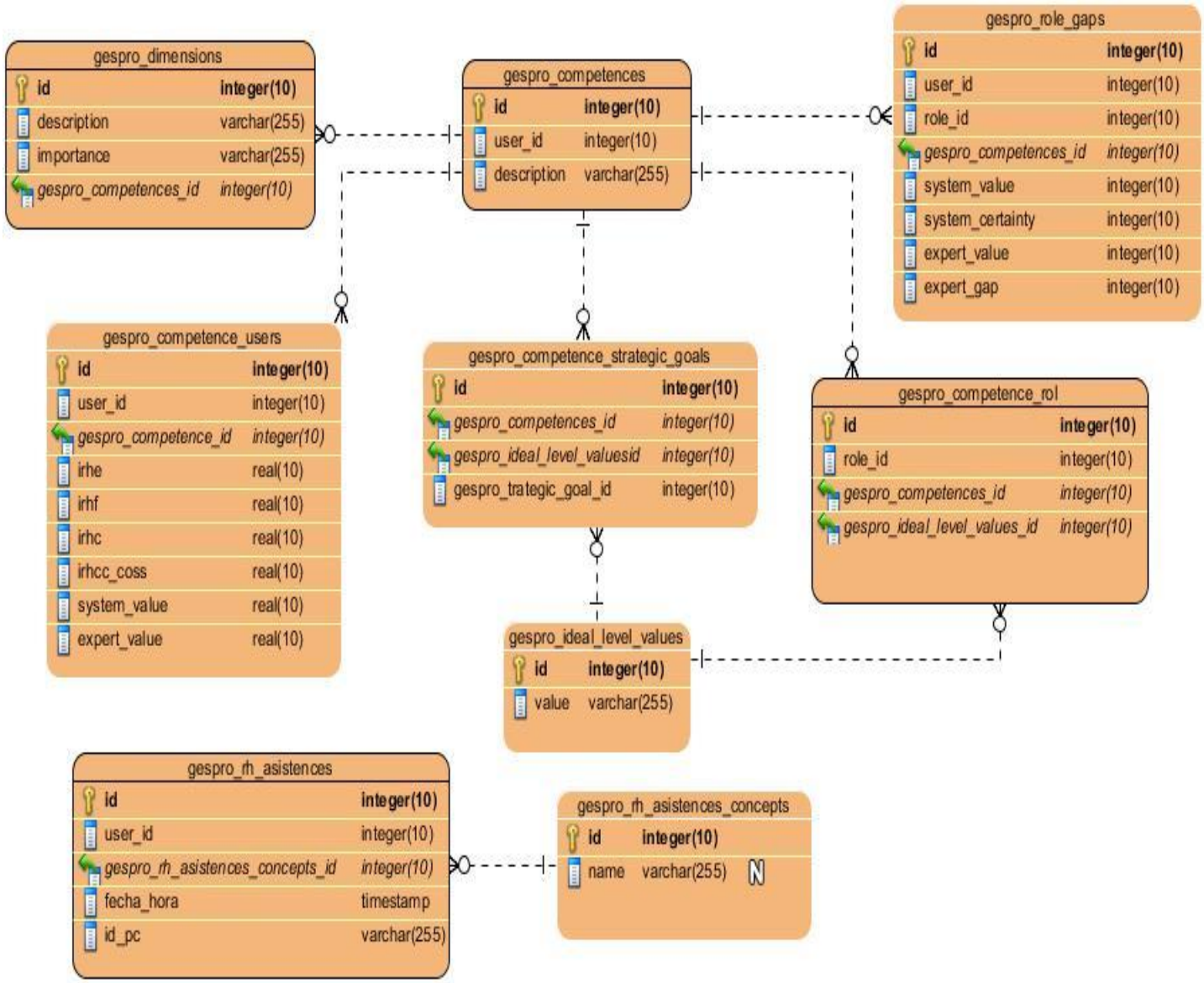


Figura 11 Diagrama entidad relación de las principales tablas del módulo.

gespro_competences*:* en esta tabla se almacenan las competencias las cuales poseen como atributos un identificador de la tabla (*id*), un nombre (*name*) y la descripción (*description*) de las mismas.

***gespro_dimensions**:* en esta tabla se almacenan las dimensiones asociadas a las competencias las cuales poseen como atributo un identificador de la tabla (*id*), la descripción (*description*) y la importancia (*importance*) de la misma.

***gespro_competence_users**:* en esta tabla se almacenan las evaluaciones de los usuarios al realizar una competencia laboral. Las mismas poseen como atributos un identificador de la tabla (*id*), el identificador del usuario (*user_id*), el identificador de la competencia (*gespro_competences_id*), y los indicadores para realizar las evaluaciones según la eficacia (*irhe*), la eficiencia (*irhf*), los costos (*irhc*) y la consistencia del conjunto (*irhcc_coss*), además de la evaluación del sistema (*system_value*) y la evaluación del experto (*expert_value*).

***gespro_competence_rols**:* en esta tabla se almacenan los niveles ideales de los roles necesarios para realizar una competencia laboral. La misma posee como atributos un identificador de la tabla (*id*), el identificador del rol (*role_id*), el identificador de la competencia (*gespro_competences_id*) y el identificador del nivel ideal para el rol (*gespro_ideal_level_values_id*).

***gespro_ideal_level_values**:* en esta tabla se encuentran los niveles ideales. La misma posee como atributos un identificador de la tabla (*id*) y el valor (*value*) del nivel ideal para una competencia.

***gespro_competence_strategic_goals**:* en esta tabla se almacenan los objetivos estratégicos asociados a una competencia laboral. La misma posee como atributos un identificador de la tabla (*id*), el identificador de la competencia (*gespro_competences_id*), el identificador del nivel ideal (*gespro_ideal_level_values_id*) y el identificador del objetivo estratégico (*gespro_strategic_goal*).

gespro_role_gaps: la misma posee como atributos el identificador de la tabla (id), el identificador del usuario (*user_id*), el identificador del rol (*role_id*), el identificador de la competencia (*gespro_competences_id*), la evaluación del sistema (*system_value*), la evaluación de certeza del sistema (*system_certainty*), la evaluación del experto (*expert_value*) y la evaluación del experto según criterio Gap (*expert_gap*).

gespro_rh_asistences_concepts: en esta tabla se encuentra la entrada/salida para realizar la el control de asistencias. La misma posee dos atributos: el identificador de la tabla (*id*) y la entrada/salida (*name*).

gespro_rh_asistences: esta tabla almacena la asistencia del usuario, posee como atributos un identificador de la tabla (*id*), fecha-hora (*fecha_hora*) y la dirección IP de la PC (*id_pc*).

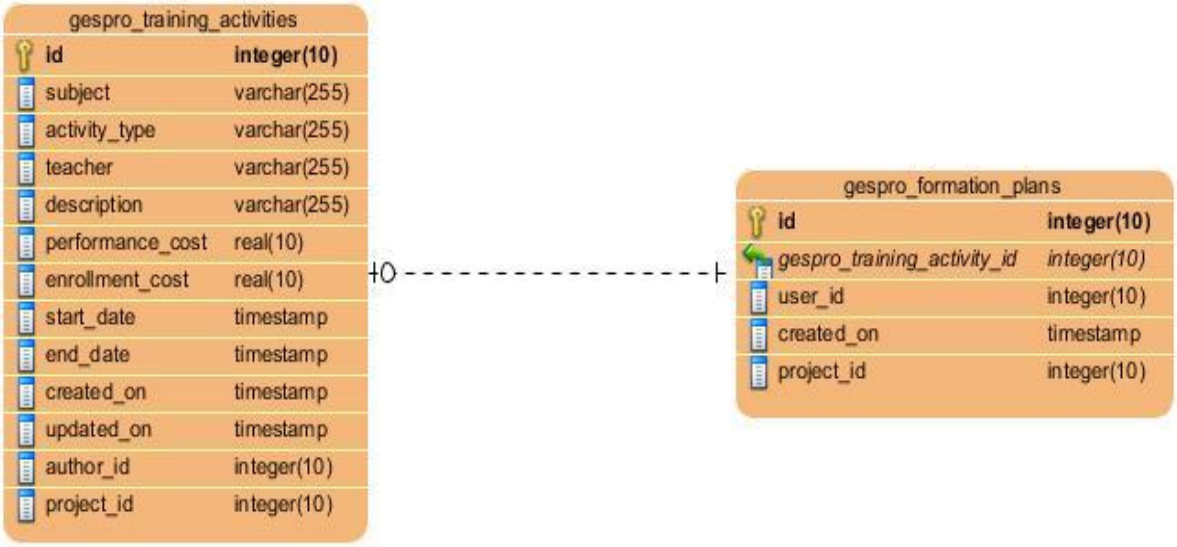


Figura 12 Diagrama entidad relación de la tabla a agregar al módulo.

gespro_training_activities: en esta tabla se almacena la información proveniente de una actividad de formación. La misma contiene como atributos un identificador de la tabla (*id*), el tema (*subject*), tipo de actividad (*activity_type*), el profesor encargado (*teacher*), la descripción (*description*), costo por impartir la actividad (*performance_cost*), el costo de matrícula (*enrollment_cost*), fecha de inicio (*start_date*) y de fin (*end_date*), la fecha en que fue creada (*created_on*) o modificada (*update_on*), el identificador del creador (*autor_id*) y el identificador del proyecto donde fue creada (p*roject_id*).

gespro_formation_plans: en esta tabla se almacena la información generada al crear un plan de formación. La misma contiene como atributos un identificador de la tabla (*id*), el identificador de la actividad de formación (*gespro_training_activity_id*), el identificador del usuario (*user_id*), la fecha de creado el plan (*created_on*) y el identificador del proyecto en el cual se creó el plan (*project_id*).

2.4.5 Diagrama de componentes

El diagrama de componentes permite visualizar con facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. En ellos se muestran los elementos de diseño de un sistema de *software* y sus dependencias.

En el diagrama de componentes del módulo de Recursos Humanos se agruparon los componentes comunes en paquetes, y se definió la relación entre estos. De esta manera, todos los componentes controladores están dentro del paquete "*Controller"*, los componentes de interfaz están dentro del paquete "*Views"* y los

componentes modelos están dentro del paquete "*Model*". Además, se representan componentes externos como las clases de estilos, las clases JavaScript y la base datos. La figura 13 muestra cómo se encuentran organizados los componentes del módulo.

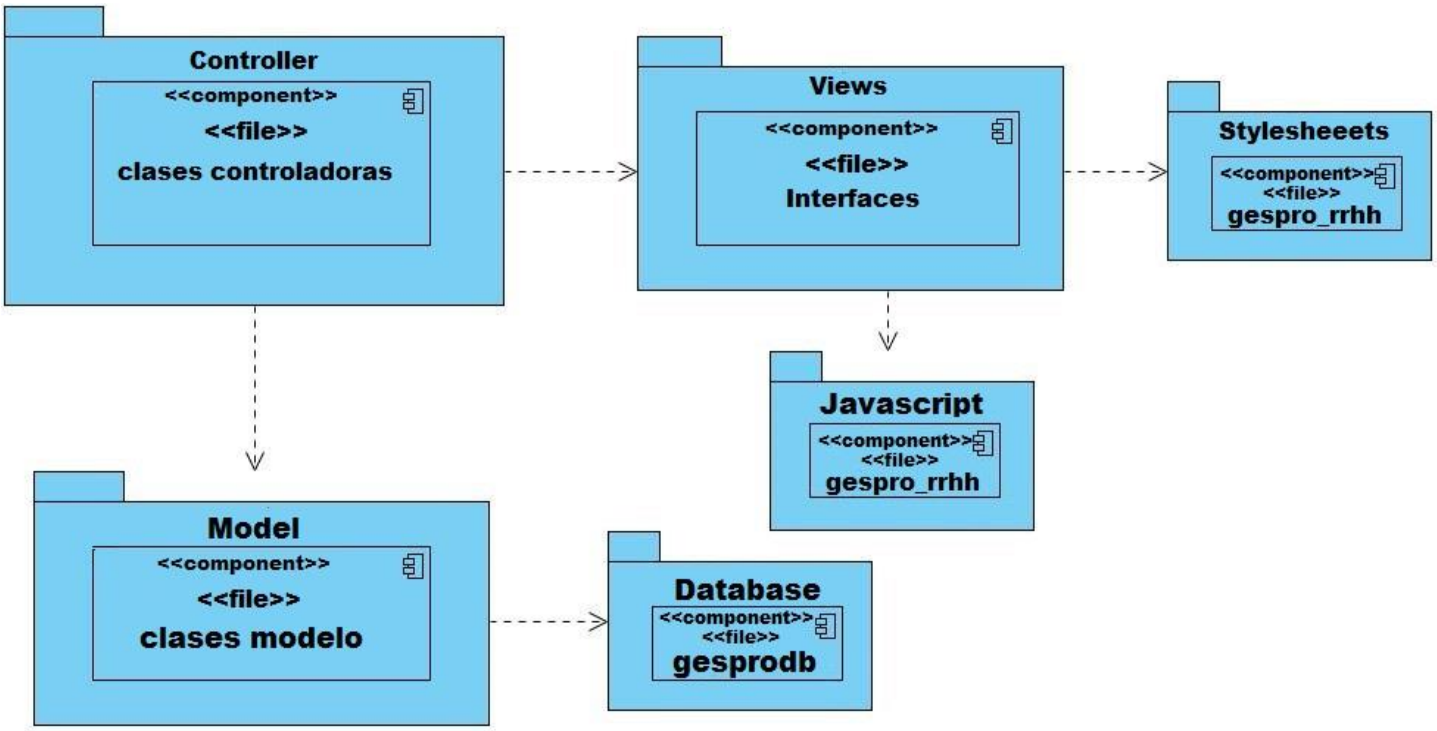


Figura 13 Diagrama de componentes del módulo de Recursos Humanos.

Conclusiones parciales

La descripción en lenguaje natural de las necesidades del cliente, a partir de la aplicación de entrevistas, posibilitó la especificación de los requisitos, funcionales y no funcionales, identificando así las funciones que debe realizar el módulo y las restricciones que este debe cumplir.

La utilización del patrón arquitectónico MVC permitió estructurar los componentes del módulo, de esta manera, se obtuvo una clara separación entre interfaz, lógica de negocio y datos, lo cual sirvió de soporte a la reutilización de los componentes reduciendo el costo y tiempo de desarrollo.

La definición de los patrones, tanto arquitectónico, de diseño y acceso a datos a utilizar permitió establecer la base estructural de la implementación del módulo.

El nuevo diseño propuesto para el módulo de Recursos Humanos garantizó incorporar al mismo un conjunto de funcionalidades que darán apoyo a la gestión de los planes de formación.

Capítulo 3: Implementación y prueba

Introducción

En el presente capítulo se documenta la fase de desarrollo con los artefactos que la misma genera según la metodología previamente seleccionada. Se expone el diagrama para el despliegue del sistema a implementar Se determina la validez de la propuesta desarrollada a través de la aplicación de pruebas al módulo y los resultados que estas generan con el propósito de verificar si los requisitos identificados realmente definen el sistema que se debe construir.

3.1 Implementación del módulo

Dentro del ciclo de vida de un sistema informático se encuentra la fase de implementación, muy costosa en cuanto a tiempo y recursos ya que se ponen en práctica todas las descripciones y arquitecturas propuestas en la fase de análisis y diseño. Se puede entender la misma como el complemento del trabajo de las fases anteriores dentro del proceso unificado de *software* en la que se completa el trabajo realizado previamente, se materializan los requisitos y se obtiene como resultado componentes de código que se compilan e integran en versiones ejecutables.

3.1.1 Estructura del módulo

El módulo de Recursos Humanos se encuentra estructurado de acuerdo a la siguiente distribución:

- ***app*:** Contiene el núcleo del módulo dividido en tres subdirectorios "*controllers"*, "*models"*, "*views" y "helpers"*.
    - ***controllers*:** Contiene las clases controladoras del módulo.
    - ***models*:** Contiene las clases de modelación de datos almacenados en la base de datos de la aplicación.
    - ***views*:** Contiene las plantillas de visualización que se llenarán con los datos de la aplicación, las mismas que serán convertidas en HTML y devueltas al navegador del usuario.
    - ***helpers:*** Contiene las clases auxiliares que proporcionan funcionalidades complementarias.
- ***assets*:** Contiene el subdirectorio "*stylesheets" y "javascripts"*.

- ***stylesheets*:** Contienen las hojas de estilo en cascada, las mismas proveen de estilos visuales a las vistas del módulo.

- ***Javascripts:*** Contiene las funciones JavaScript.

- ***config:*** Contiene el subdirectorio "*locales*" y el archivo "*routes.rb"*.

  - ***locales*:** Se almacenan los archivos dedicados a los idiomas de la aplicación permitiendo la fácil migración a cualquiera de estos. Actualmente el módulo brinda soporte para inglés y español.
  - ***routes:*** Se especifican las rutas para acceder a las interfaces.

- ***db*:** Contiene el subdirectorio "*migrate"*.

  - ***migrate*:** Contiene archivos SQL para la creación y modificación de tablas en la base de datos. Los *migrates* permiten trabajar independiente del servidor central, y una vez terminado, es posible migrar la base de datos central.

- ***lib:*** Contiene los subdirectorios "*Math*' y '*pfbcl'*.

  - ***pfbcl:*** Contiene el parche de la clase controladora "*ProjectAplicationController".*
  - ***Math:*** Contiene la clase "*Plan"* donde se implementa un algoritmo para proponer planes de formación.

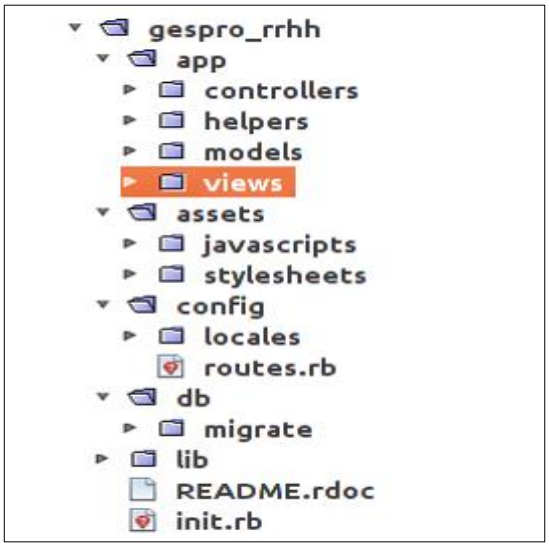


Figura 14 Estructura del módulo.

3.1.2 Descripción de las clases

Tabla 4 Descripción de las clases controladoras del módulo.

Clases controladoras (app/controller)	
BaseRrhhApplicationController	Clase padre de todas las clases controladoras. Contiene todos los métodos genéricos heredados por otras clases.
GesproCompetencesController	Manipula la información concerniente a las competencias laborales.
GesproDimensionsController	Manipula la información concerniente a las dimensiones asociadas a las competencias.
GesproRhProjectAsistencesController	Registra y muestra las asistencias de los usuarios a nivel de proyecto.
GesproRrhhProjectController	Encargada del control de los eventos del módulo a nivel de proyecto.
GesproCompetenceRolsController	Maneja la información de los roles asociados a cada competencia.
GesproCompetenceUsersController	Maneja la información de los usuarios asociados a cada competencia.
GesproTrainingsController	Controla la información de las actividades de formación.

Tabla 5 Descripción de las clases modelos del módulo.

Clases modelo (app/modelo)	
GesproTrainingActivity	Permite la conexión directa con la tabla *gespro_training_activities*, que representa las actividades de formación.

GesproCompetence	Permite la conexión directa con la tabla *gespro_competences*, que representa las competencias laborales.
GesproCompetenceRol	Permite la conexión directa con la tabla *gespro_competence_rols*, que representa las competencias laborales asociadas a un rol o perfil.
GesproCompetenceUser	Permite la conexión directa con la tabla *gespro_competence_users*, que representa las competencias laborales asociadas a los usuarios del sistema.
GesproDimension	Permite la conexión directa con la tabla *gespro_dimensions*, que representa las dimensiones de las competencias laborales.
GesproRhAsistence	Permite la conexión directa con la tabla *gespro_rh_asistences*, que representa las asistencias de los usuarios.
GesproIdealLevelValue	Permite la conexión directa con la tabla *gespro_ideal_level_values*, que representa los niveles ideales de las competencias laborales.

Tabla 6 Descripción de las clases auxiliares del módulo.

Clases auxiliares (…/…)	
Plan	Implementa un algoritmo que permite, según los niveles del usuario al realizar una competencia, el impacto de las actividades de formación sobre la competencia, además de verificar el importe máximo y el presupuesto, proponer una actividad de formación para el usuario.

GesproCompetenceHelper	Contiene métodos auxiliares para las funcionalidades asociadas a las competencias laborales.
GesproDimensionHelper	Provee al sistema de métodos auxiliares asociados a las dimensiones.
TrainingsHelper	Provee un método auxiliar para realizar la paginación al listar las actividades de formación.
ProjectsHelperPatchTrainings	Provee de métodos auxiliares para obtener los niveles de los usuarios y el impacto de las actividades sobre las competencias.
ProjectsControllerPatch	Controla los eventos del módulo referente a las actividades de formación y las competencias a nivel de proyecto.

Las clases controladoras, como indica su nombre, son las encargadas de controlar todo el proceso por el cual transcurrirá el negocio. Una de sus principales tareas es controlar el flujo de datos generado por las peticiones de los usuarios, por lo cual da instrucciones al modelo para su cumplimiento.

Las clases modelos son las encargadas de la comunicación con la base de datos. Cada una representa una tabla de la base de datos cuyos atributos son representados como las propiedades de la clase y una instancia representa una fila en esa tabla.

Para agregar dichas tablas se utilizan los archivos "*migrate**"*, los cuales son sentencias de DDL (lenguaje de definición de datos) escritas en Ruby, que nos permiten administrar el esquema de la BD. Cuando se ejecuta un "*migrate"* (migración) se cambia la versión de un esquema a otra anterior o posterior.

Las migraciones son almacenadas en archivos en el directorio ***db/migrate***, uno por cada clase. El nombre del archivo es de la forma ***YYYYMMDDHHMMSS_name_file.rb***, esto es la marca de tiempo UTC (Tiempo Universal Coordinado) identificando la migración seguida por un '_' seguido del nombre de la migración. El nombre de clase de la migración debe coincidir con la última parte del nombre de archivo. Por ejemplo ***20121025162022_create_gespro_competences.rb*** debe ser definida como ***CreateGesproCompetences***.

Ejemplo de "*migrate"* para agregar la tabla *gespro_training_activities* a la base de datos. La misma almacenará las actividades de formación:

```
class CreateGesproTrainingActivities < ActiveRecord::Migration

 def self.up

  create_table :gespro_training_activities do |t|

   t.column :subject,      :string,  :null => false, :limit => 100
   t.column :activity_type,  :string,  :null => false, :limit => 30
   t.column :teacher,      :string,  :null => false, :limit => 100
   t.column: description,   :text,   :null => false
   t.column :performance_cost, :decimal, :null => false, :default => 0, :precision => 6, :scale => 2
   t.column :enrollment_cost, :decimal, :null => false, :default => 0, :precision => 6, :scale => 2
   t.column :start_date,     :date,   :null => false
   t.column :end_date,      :date,   :null => false
   t.column :created_on,     :datetime, :null => false
   t.column :updated_on,     :datetime, :null => false
   t.column :author_id,     :integer, :null => false, :default => 0
   t.column :project_id,     :integer, :null => false, :default => 0

 end

end

def self.down

 drop_table :training_activities

end

end
```

Este "*migrate"* agrega una tabla llamada *gespro_training_activities* con doce columnas, de ellas, tres de tipo "*string"* de nombre *subject, activity_type* y *teacher*, una de tipo "*text"* llamada *description*, dos de tipo "*decimal"* de nombres *performance_cost* y *enrollment_cost*, otras dos de tipo "*date"* de nombre *start_date* y *end_date*,

dos más de tipo "*datetime"* nombradas *created_on* y *updated_on*, y por último dos de tipo "*integer"* llamadas *autor_id* y *project_id*. Además, una columna llave primaria llamada *id* es creada por omisión.

3.1.3 Diagrama del despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados. Se muestran las relaciones físicas entre los componentes *hardware* y *software* en una aplicación. Cada *hardware* (máquinas físicas y los procesadores) se representa como un nodo (Valdivia Cerda, 2003).

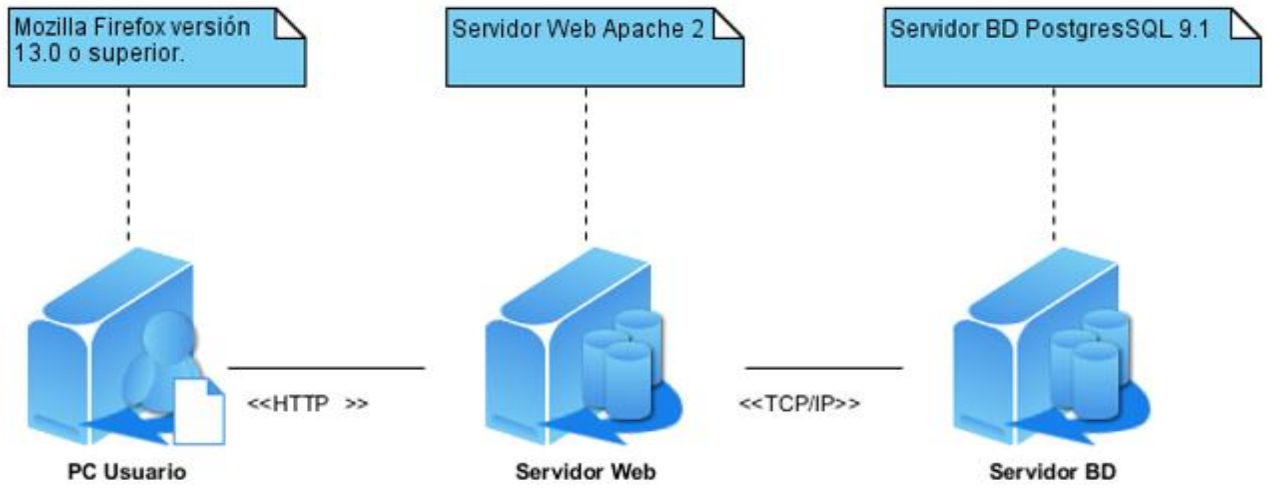


Figura 15 Diagrama de despliegue del módulo de Recursos Humanos.

El nodo Servidor BD está conectado con el nodo Servidor Web a través del protocolo de red TCP/IP, este a su vez está conectado con los nodos clientes a través de protocolos de comunicación HTTPS, garantizando que en cada estación de trabajo los usuarios tengan acceso a la herramienta de manera segura. XEDRO GESPRO está instalado en un servidor web Apache, el mismo, utiliza *PostgreSQL* como sistema gestor de base de datos.

3.2 Prueba

Las pruebas de *software* son elementos imprescindibles y críticos para la validación de un producto de *software*. Por esta razón, tales pruebas deben apoyarse en estándares que revisan los aspectos fundamentales que debe considerar todo proceso de pruebas.

El proceso de pruebas se dirige fundamentalmente a componentes del *software* o al sistema de *software* en general, con el objetivo de medir hasta cuando el mismo cumple las funcionalidades solicitadas por el cliente.

Las pruebas del *software* verifican y revelan la calidad de un producto. Son utilizadas para identificar posibles errores en la implementación, calidad, o usabilidad de un programa de *software*.

3.2.1 Validación de las nuevas funcionalidades a través de las pruebas de caja negra

Las pruebas funcionales de caja negra se refieren a las pruebas que se llevan a cabo sobre la interfaz del *software*, por tal motivo, los casos de prueba pretenden demostrar que las funciones del propio *software* son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene.

Conociendo las funciones específicas para la que fue rediseñado el producto, se pueden diseñar un conjunto de pruebas que demuestren que dichas funciones están bien realizadas. Tales pruebas son llevadas a cabo sobre las interfaces del módulo, es decir, de las funciones actuando sobre ellas como una caja negra, proporcionando unas entradas y estudiando las salidas para verificar que las mismas concuerdan con las esperadas.

A continuación se muestran los resultados obtenidos a partir de la aplicación de pruebas funcionales de caja negra a una de las nuevas funcionalidades con la que cuenta el módulo. El resto de las pruebas aplicadas se encuentran en la carpeta Anexos adjunta a la investigación. Ver Anexo 2 y Anexo 3.

Funcionalidad 2 Gestionar actividades de formación:

- **Escenario 1: Crear actividad de formación de forma correcta.**

En la figura 16 se muestra el diseño de caso de prueba para la funcionalidad 2.1 crear actividad de formación, así como se muestra en la figura 17 la descripción de las variables necesarias para ejecutar este caso de prueba y en las figuras 18, 19 y 20 se visualizan las interfaces que muestran el flujo del proceso para crear una actividad de formación.

Escenario	Descripción	Asunto	Tipo de actividad	Nombre del profesor	Descripción	Fecha inicio	Fecha fin	Costo de realización	Costo de matriculación	Respuesta del sistema
EC 1 Crear una actividad de formación de manera correcta	El sistema permite crear una nueva actividad de formación de forma correcta.	V Ingresar nombre de la actividad	V elelgir tipo de actividad	Ingresar el nombre del profesor que impartira la actividad	V Ingresar una descripción para la actividad	V elegir fecha de inicio	V elegir fecha de fin	V ingresar costo para la realización de la actividad	V ingresar costo para matricular en la actividad	Se muestra mensaje de éxito. Redirecciona a la vista de impacto.
EC 2 Crear una actividad de formación de manera incorrecta	El sistema no permite crear una nueva actividad de formación porque alguna/s de las variables no existen o no tienen el formato correspondiente.	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	Se muestra mensaje indicando el o los campos vacios. Se mantiene la vista actual.
		V Ingresar nombre de la actividad	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	
		I (vacio)	V elelgir tipo de actividad	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	
		I (vacio)	I (vacio)	V Ingresar el nombre del profesor que impartira la actividad	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	
		I (vacio)	I (vacio)	I (vacio)	V Ingresar una descripción para la actividad	I (vacio)	I (vacio)	I (vacio)	I (vacio)	
		I (vacio)	I (vacio)	I (vacio)	I (vacio)	V elegir fecha de inicio	I (vacio)	I (vacio)	I (vacio)	
		I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	V elegir fecha de fin	I (vacio)	I (vacio)	
		I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	V ingresar costo para la realización de la actividad	I (vacio)	
		I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	V ingresar costo para matricular en la actividad	

Figura 16 Diseño del caso de prueba crear actividad de formación.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Asunto	Campo de Texto	No	Nombre de la actividad
2	Tipo de actividad	Lista Desplegable	No	Tipo de actividad
3	Nombre del profesor	Campo de Texto	No	Nombre del profesor encargado la actividad
4	Descripción	Campo de Texto	No	Descripción de la actividad
5	Fecha inicio	calendario	No	Fecha en la cual dará comienzo la actividad
6	Fecha fin	calendario	No	Fecha en la cual culminará la actividad
7	Costo de realización	valor n umérico	No	Valor monetario para realizar la actividad
8	Costo de matriculación	valor n umérico	No	Valor monetario para matricular en una actividad

Figura 17 Descripción de variables.

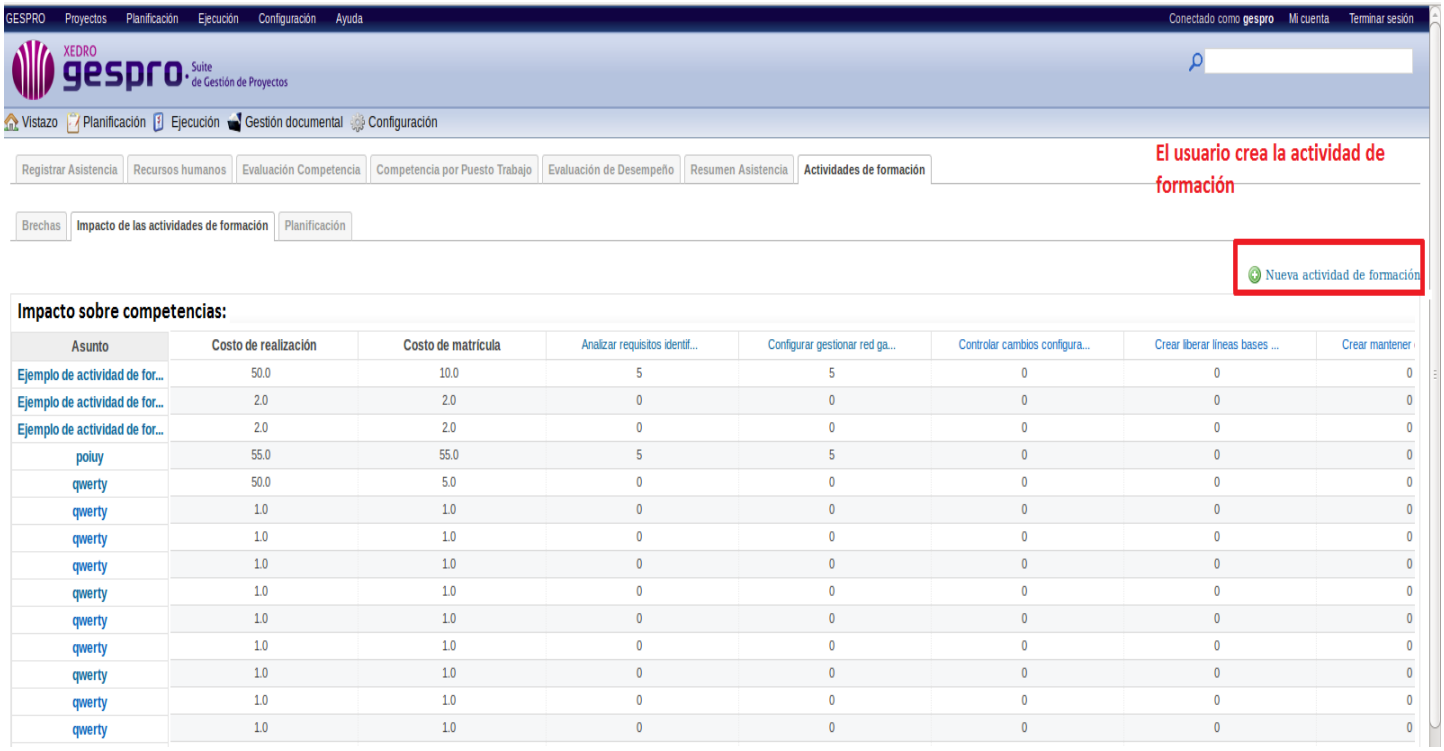


Figura 18 Crear actividad de formación.

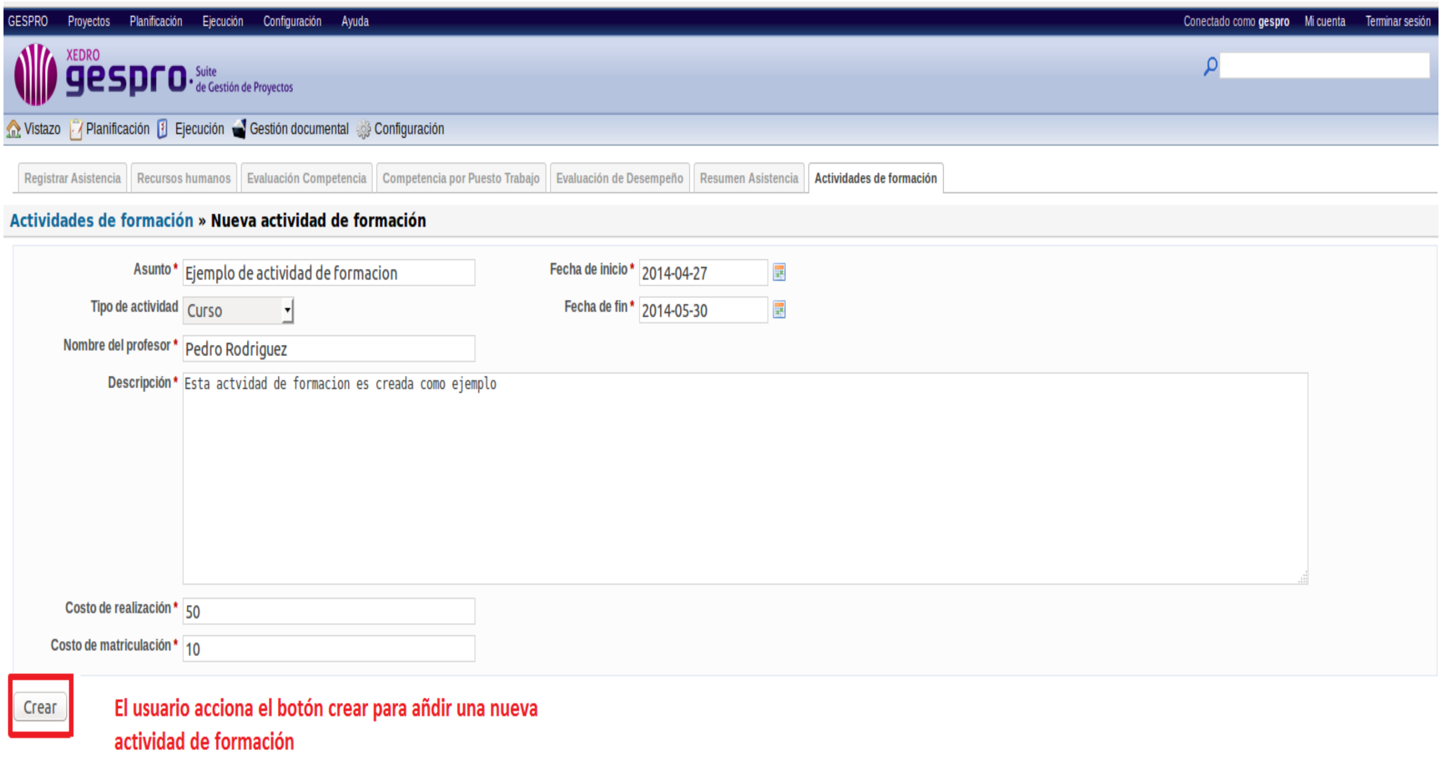
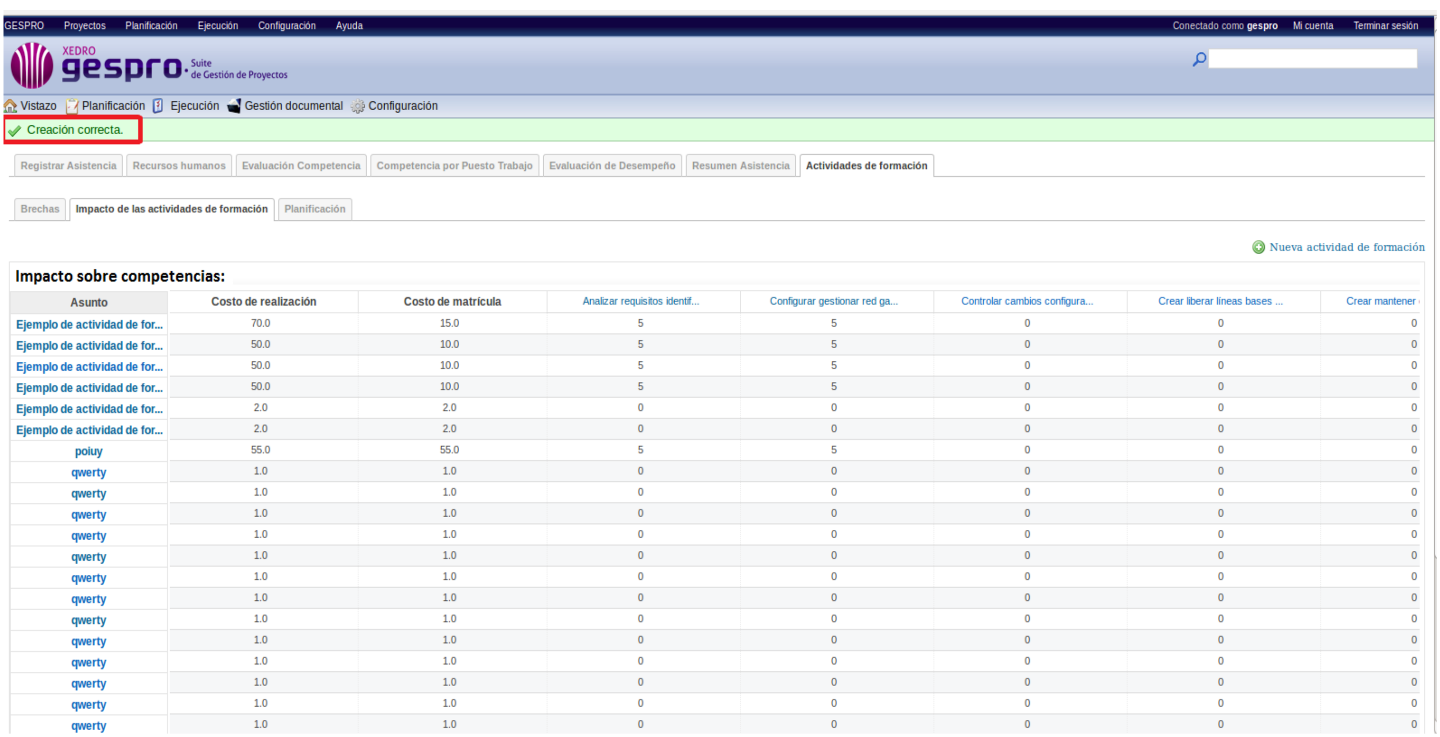


Figura 19 Formulario para crear una actividad deformación.



Asunto	Costo de realización	Costo de matrícula	Analizar requisitos identif...	Configurar gestionar red ga...	Controlar cambios configura...	Crear liberar líneas bases ...	Crear mantener
Ejemplo de actividad de for...	70.0	15.0	5	5	0	0	0
Ejemplo de actividad de for...	50.0	10.0	5	5	0	0	0
Ejemplo de actividad de for...	50.0	10.0	5	5	0	0	0
Ejemplo de actividad de for...	50.0	10.0	5	5	0	0	0
Ejemplo de actividad de for...	2.0	2.0	0	0	0	0	0
Ejemplo de actividad de for...	2.0	2.0	0	0	0	0	0
poiuy	55.0	55.0	5	5	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0
qwerty	1.0	1.0	0	0	0	0	0

Figura 20 Respuesta del sistema.

- **Escenario 1.1 Crear actividad de formación dejando campos en blanco.**

La figura 21 muestra la respuesta del sistema a la acción de crear una actividad de formación dejando los campos en blancos.

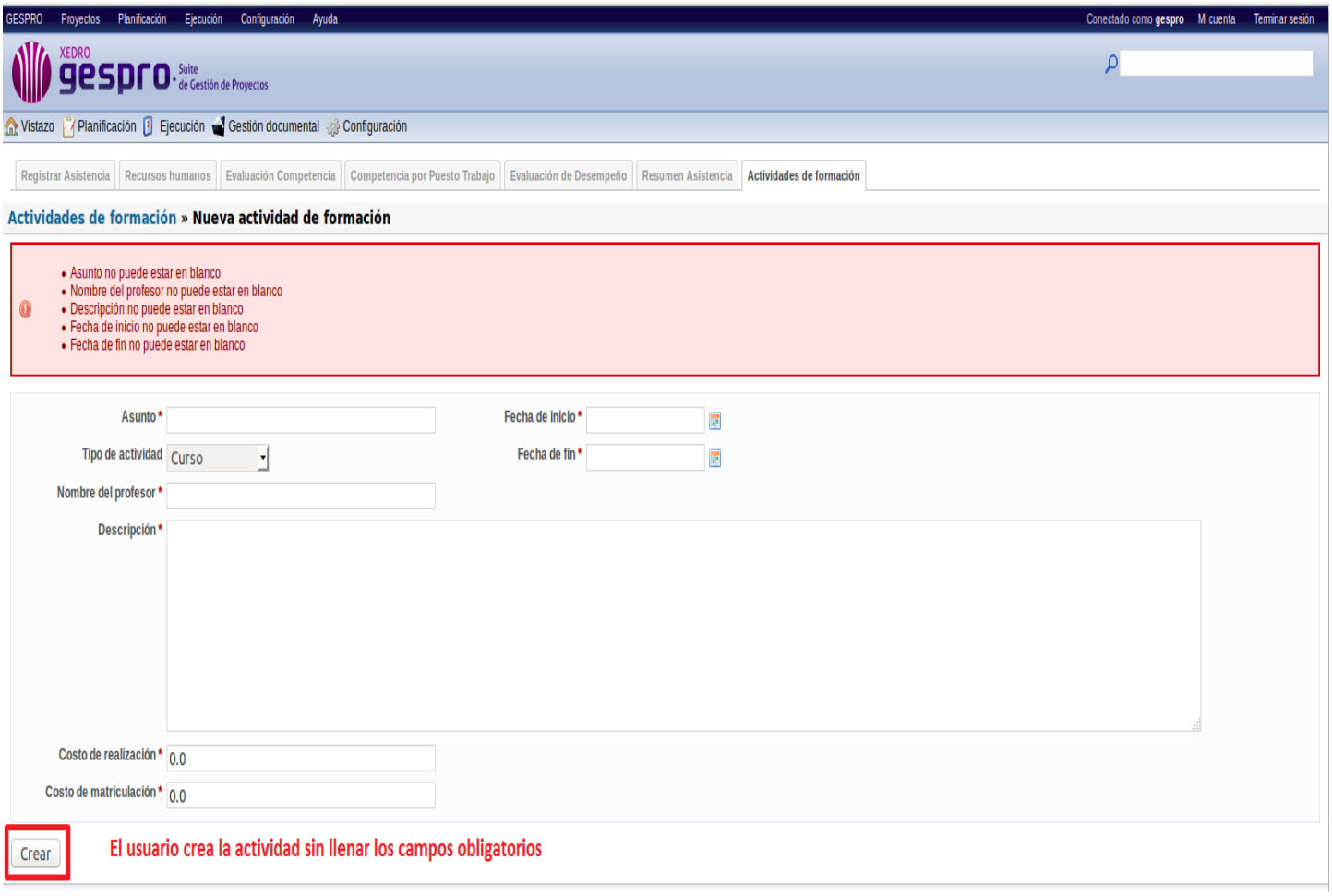


Figura 21 Alerta del sistema al dejar campos obligatorios sin llenar.

- **Escenario 1: Modificar actividad de formación.**

En la figura 22 se muestra el diseño de caso de prueba para la funcionalidad 2.2 modificar actividad de formación, así como se muestra en la figura 23 la descripción de las variables necesarias para ejecutar este caso de prueba y en las figuras 24, 25 y 26 se visualizan las interfaces que muestran el flujo del proceso para modificar una actividad de formación.

Escenario	Descripción	Asunto	Tipo de actividad	Nombre del profesor	Descripción	Fecha inicio	Fecha fin	Costo de realización	Costo de matriculación	Respuesta del sistema
EC 1 Modificar una actividad de formación de manera correcta	El sistema permite modificar una nueva actividad de formación de forma correcta.	V Ingresar nombre de la actividad	V elelgir tipo de actividad	 Ingresar el nombre del profesor que impartira la actividad	V Ingresar una descripción para la actividad	V elegir fecha de inicio	V elegir fecha de fin	V ingresar costo para la realización de la actividad	V ingresar costo para matricular en la actividad	Se muestra mensaje de éxito. Redirecciona a la vista de impacto.
EC 2 Modificar una actividad de formación de manera incorrecta	El sistema no permite modificar la actividad de formación porque alguna/s de las variables no existen o no tienen el formato correspondiente.	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	Se muestra mensaje indicando el o los campos vacios. Se mantiene la vista actual.
		V Ingresar nombre de la actividad	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	
		I (vacio)	V elelgir tipo de actividad	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	
		I (vacio)	I (vacio)	V Ingresar el nombre del profesor que impartira la actividad	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	
		I (vacio)	I (vacio)	I (vacio)	V Ingresar una descripción para la actividad	I (vacio)	I (vacio)	I (vacio)	I (vacio)	
		I (vacio)	I (vacio)	I (vacio)	I (vacio)	V elegir fecha de inicio	I (vacio)	I (vacio)	I (vacio)	
		I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	V elegir fecha de fin	I (vacio)	I (vacio)	
		I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	V ingresar costo para la realización de la actividad	I (vacio)	
		I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	I (vacio)	V ingresar costo para matricular en la actividad	

Figura 22 Diseño del caso de prueba modificar actividad de formación.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Asunto	Campo de Texto	No	Nombre de la actividad
2	Tipo de actividad	Lista Desplegable	No	Tipo de actividad
3	Nombre del profesor	Campo de Texto	No	Nombre del profesor encargado la actividad
4	Descripción	Campo de Texto	No	Descripción de la actividad
5	Fecha inicio	calendario	No	Fecha en la cual dará comienzo la actividad
6	Fecha fin	calendario	No	Fecha en la cual culminará la actividad
7	Costo de realización	valor n umérico	No	Valor monetario para realizar la actividad
8	Costo de matriculación	valor n umérico	No	Valor monetario para matricular en una actividad

Figura 23 Descripción de variables.

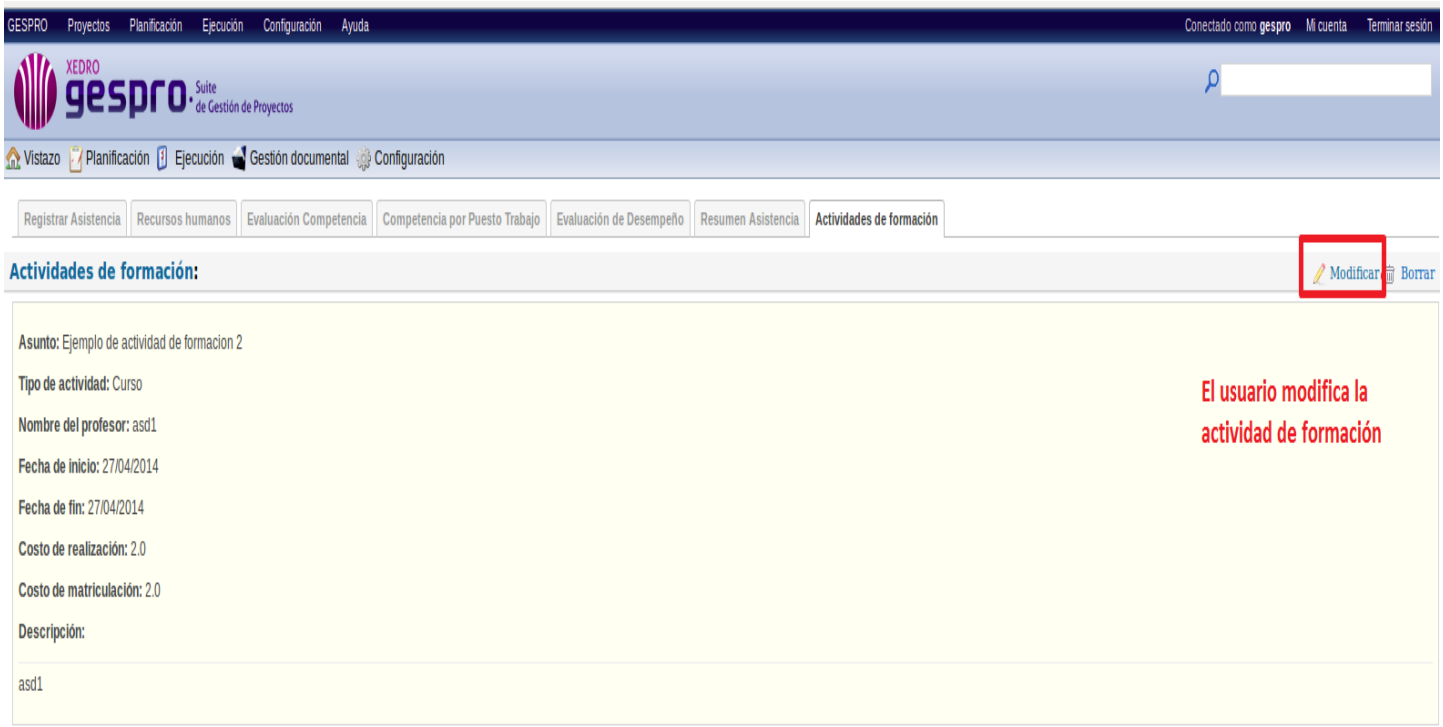


Figura 24 Modificar actividad de formación.

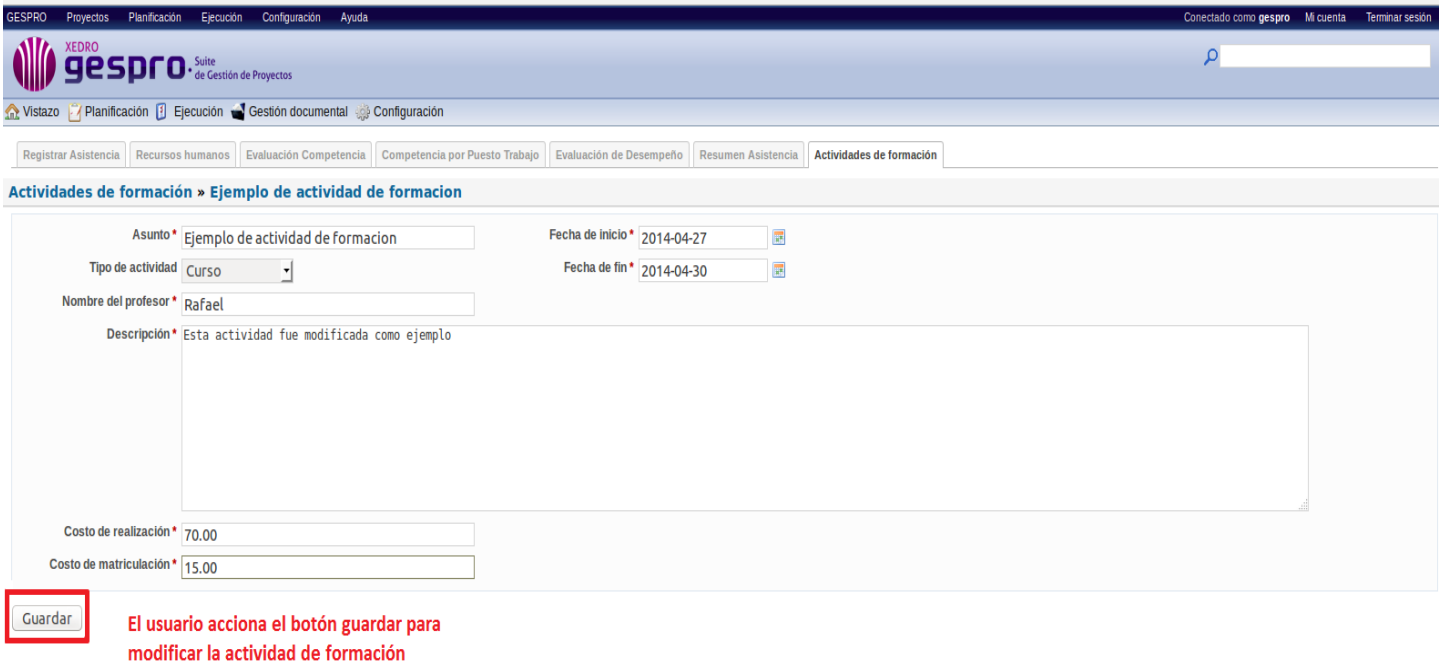


Figura 25 Formulario para modificar una actividad de formación.

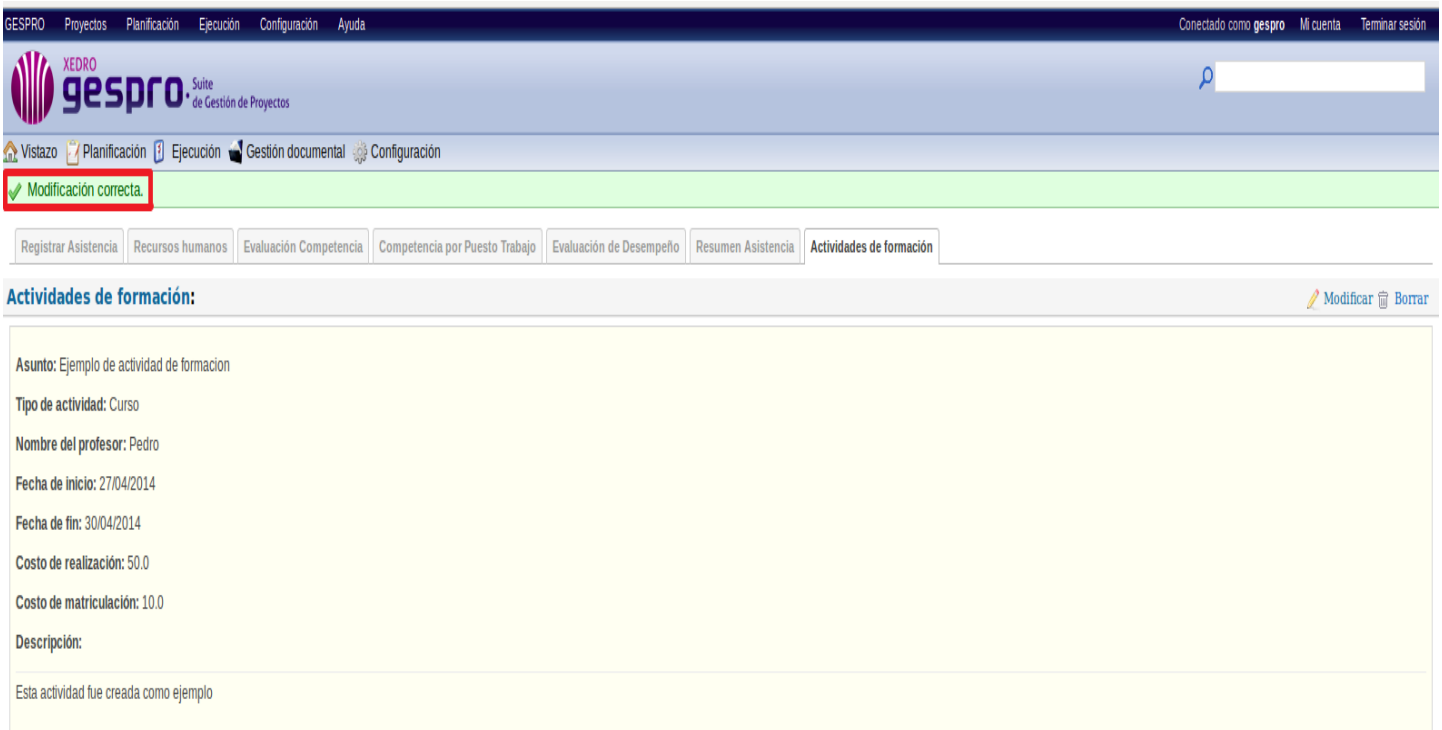


Figura 26 Respuesta del sistema.

- **Escenario 3: Eliminar la actividad de formación.**

En la figura 27 se muestra el diseño de caso de prueba para la funcionalidad 2.3 eliminar actividad de formación y en las figuras 28, 29 y 30 se visualizan las interfaces que muestran el flujo del proceso para eliminar una actividad de formación.

Escenario	Descripción	Confirmación del sistema	Respuesta del sistema
EC 1 Eliminar una actividad de formación.	El sistema permite eliminar una actividad de formación de forma correcta.	Se muestra un cartel para confirmar la eliminación de la actividad.	Se muestra mensaje de éxito. Redirecciona a la vista de impacto.

Figura 27 Diseño del caso de prueba eliminar actividad de formación.

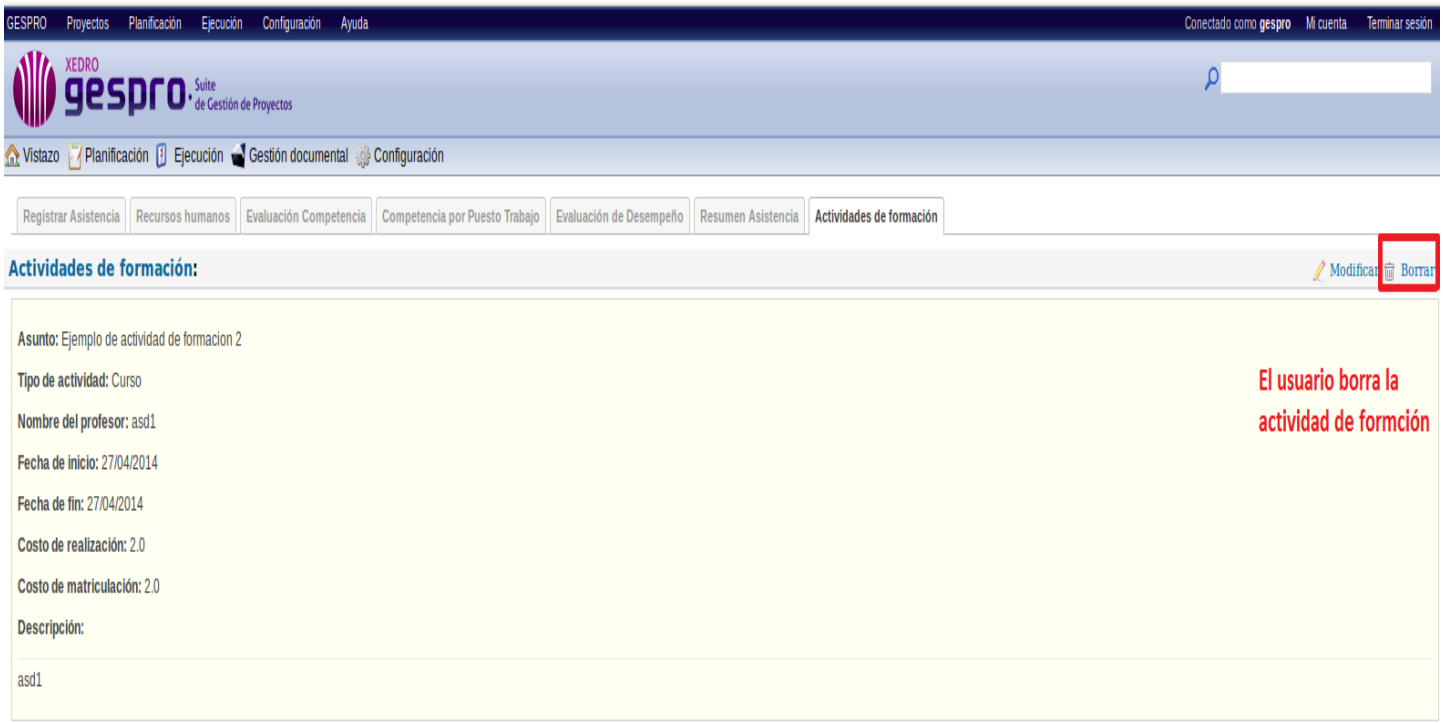


Figura 28 Eliminar actividad de formación.

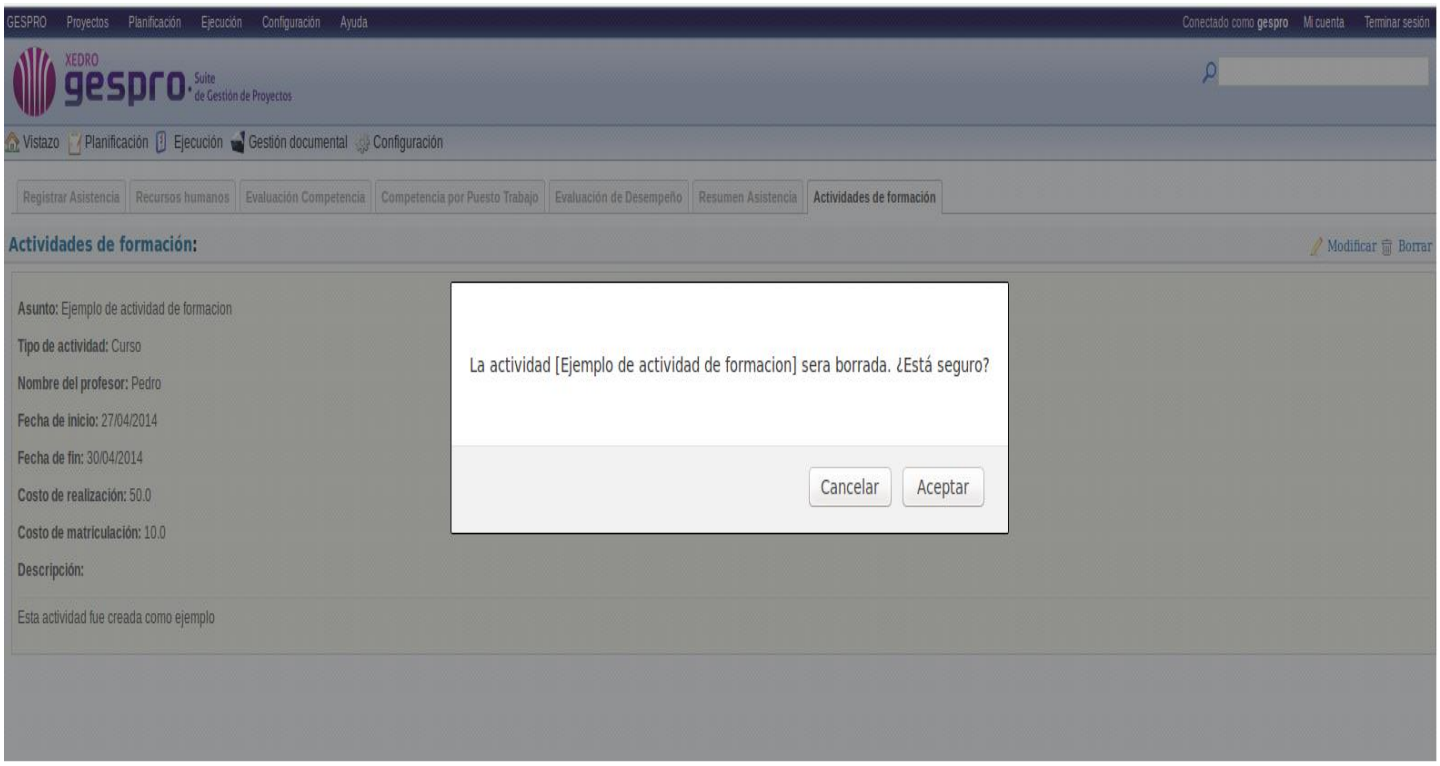


Figura 29 Confirmación para la acción eliminar.

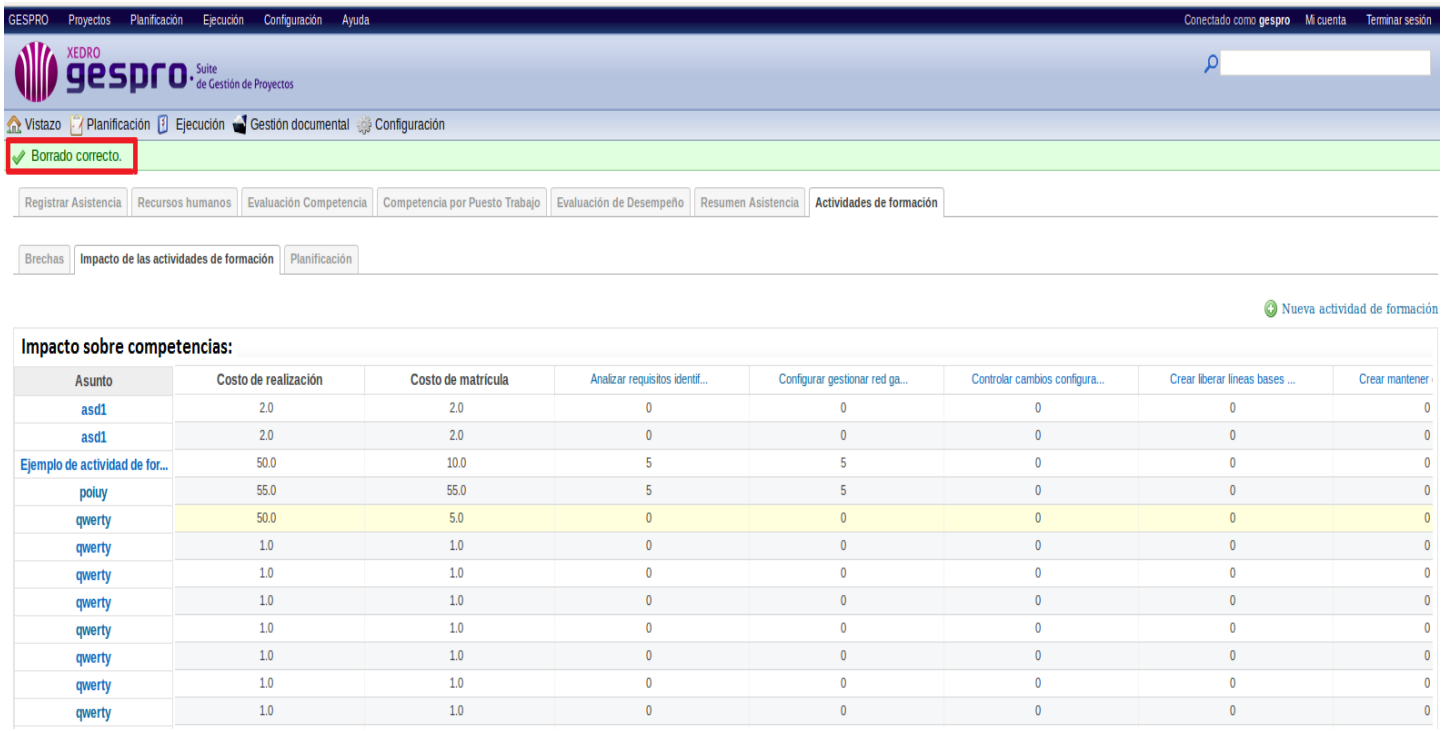


Figura 30 Respuesta del sistema.

- **Escenario 4: Mostrar la información de una actividad de formación.**

En la figura 31 se muestra el diseño de caso de prueba para la funcionalidad 2.4mostrar actividad de formación y en las figuras 32, y 33 se visualizan las interfaces que muestran el flujo del proceso para mostrar la información una actividad de formación.

Escenario	Descripción	Respuesta del sistema
EC 1 Mostrar información de una actividad de formación.	El sistema permite mostrar la información de una actividad de formación escogida por el usuario al hacer "click" sobre ella .	Direcciona a la vista "show" donde se muestra la información de la actividad de formación escogida por el usuario.

Figura 31 Diseño del caso de prueba mostrar la información de una actividad de formación.

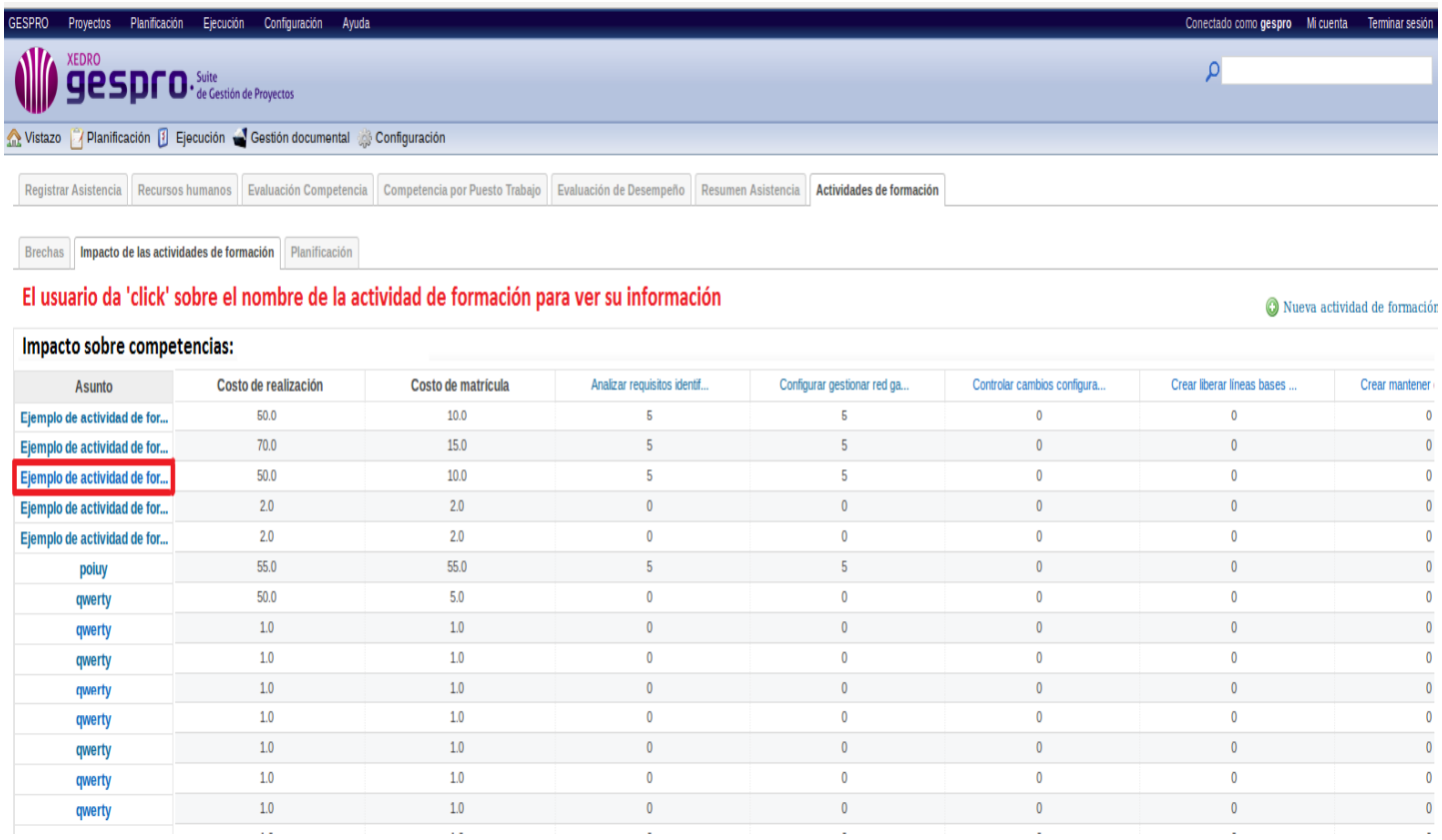


Figura 32 Mostrar actividad de formación.

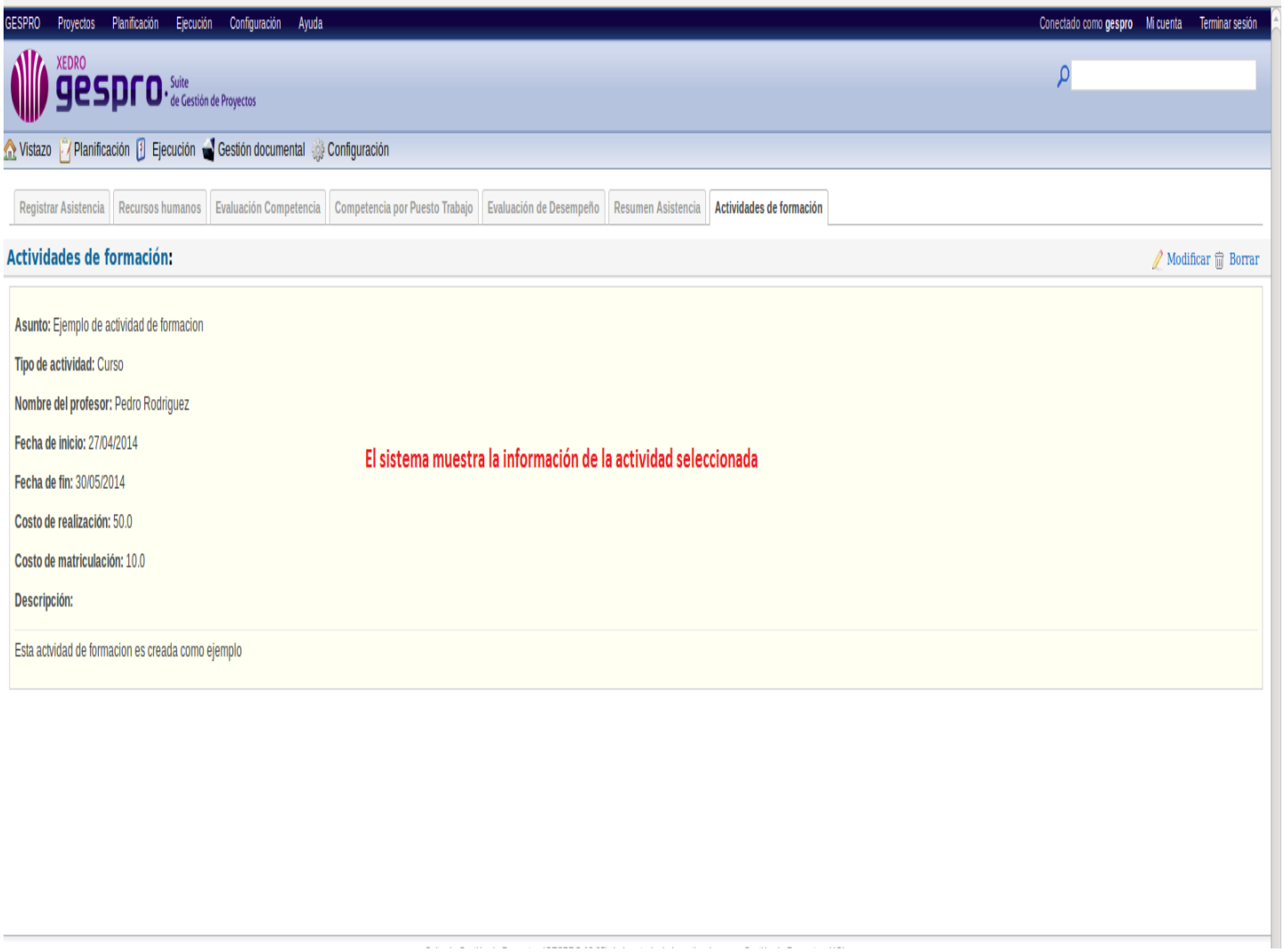
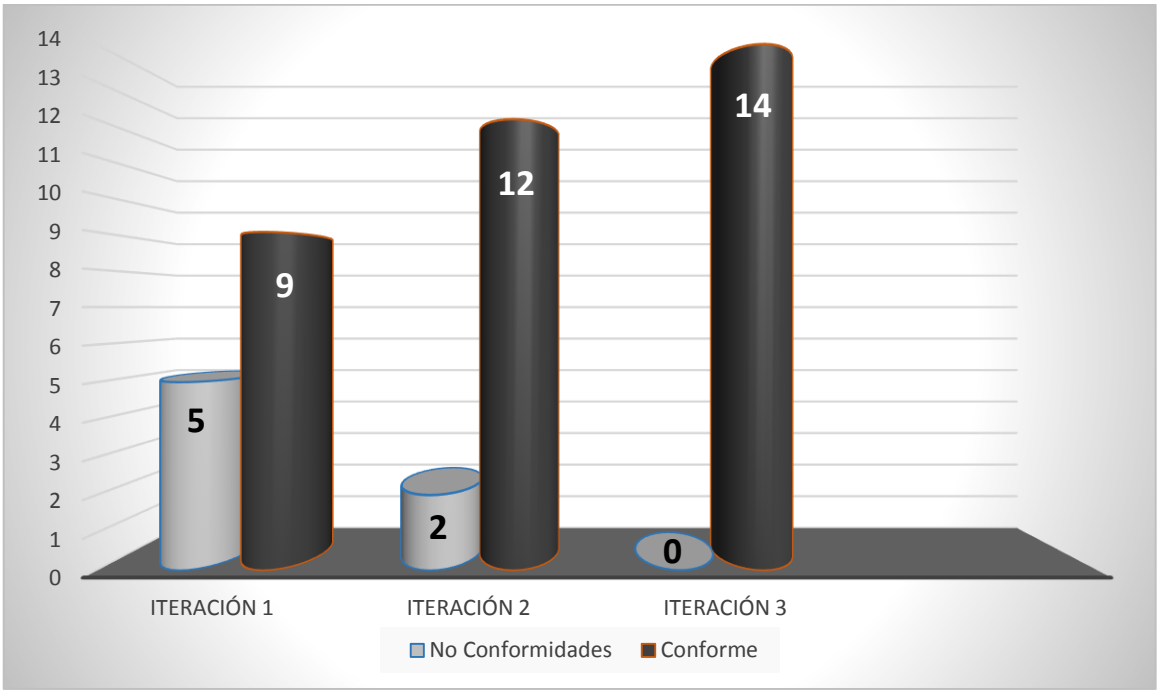


Figura 33 Mostrar información de la actividad de formación.

3.2.2 Resultados obtenidos

Después de llevadas a cabo las pruebas pertinentes para probar el correcto funcionamiento de las funcionalidades asociadas al nuevo módulo, las cuales fueron realizadas en 3 iteraciones (Ver figura 34), fue encontrada una serie de no conformidades que han sido erradicadas durante el transcurso de las mismas. En la primera iteración se detectaron 5 no conformidades, en la segunda se encontraron 2 no conformidades, estas últimas fueron completamente resueltas para en una tercera iteración tener un resultado favorable de cero no conformidades y un sistema en óptimas condiciones para su uso y manejo.



Grafica 1 Resultado de las pruebas.

3.3 Comparación de herramientas de gestión de proyectos

La Tabla 7 muestra un estudio comparativo de las principales herramientas de gestión de proyectos, con el objetivo de demostrar que con el rediseño e implementación del módulo de Recursos Humanos el sistema GESPRO, en su versión 13.05, se convierte en una herramienta más competitiva y superior en el proceso de desarrollo de los Recursos Humanos respecto a muchas de estas herramientas.

Tabla 7 Comparación de herramientas de gestión de proyectos según procesos de gestión de Recursos Humanos.

Herramientas	Planificación	Adquisición	Desarrollo	Gestión
GESPRO	Si	Si	Si	Si
dotproject	Si	Si	No	Si
MS Project 2010	Si	Si	No	Si
Microsoft Project	Si	No	No	No
OpenProject	No	Si	No	Si
Oracle	Si	Si	No	No
Redmine	Si	Si	No	Si

SAP RPM	**Si**	**Si**	**Si**	**Si**
TeamworkPM	**No**	**No**	**No**	**Si**
Achievo	**Si**	**Si**	**No**	**Si**
Collabtive	**Si**	**No**	**No**	**Si**

Como se observa en la Tabla 7, el sistema GESPRO es hoy una de las herramientas más potentes, al nivel de que puede competir con cualquiera de los líderes mundiales en la fabricación de *software* de gestión de proyectos. Dicha tabla nos refleja cómo el sistema GESPRO es una de las más completas herramientas que existen para la gestión de proyectos, en cuanto a los procesos de gestión de Recursos Humanos.

3.4 Impacto de la propuesta en GESPRO 13.05

Con el rediseño e implementación del módulo de Recursos Humanos de la herramienta GESPRO 13.05 se incluye en la misma la gestión de los planes de formación. Estas funcionalidades facilitan la formación continua del personal de los proyectos, elemento que ha sido identificado como uno de los principales retos de la gestión de Recursos Humanos a nivel internacional (Chiavenato, 2007).

En Cuba la formación continua también constituye un reto ya que muchas organizaciones se ven afectadas por la fluctuación de su personal, evidenciándose la necesidad de contar con herramientas que faciliten este proceso en correspondencia con lo planteado en los lineamientos 138 y 172 del capítulo 4 de la política económica y social del país (PCC, 2011).

Conclusiones parciales

En este capítulo se detallaron las tareas de la fase de implementación y se obtuvo como resultado un módulo funcional para la gestión de Recurso Humanos proporcionando nuevas funcionalidades asociadas a las condiciones y necesidades actuales en esta área.

Las descripciones de las clases proporcionaron un mayor entendimiento de las responsabilidades de las mismas al realizar sus respectivas funciones.

La realización de las pruebas funcionales de caja negra permitió la corrección de errores, garantizando así validar la implementación de las funcionalidades del módulo obteniendo resultados positivos para el problema de la investigación.

Conclusiones generales

Como resultado de la investigación realizada se concluye que:

- El análisis de los elementos teóricos asociados al negocio facilitó la definición de una propuesta de solución acorde a las necesidades existentes.
- El establecimiento de Scrum como metodología rectora del desarrollo del módulo, y el uso de las tecnologías y herramientas definidas, permitieron analizar, describir e implementar los procesos y subprocesos que debían ejecutarse, materializando así una solución acorde a los requerimientos del cliente.
- La definición e implementación de las funcionalidades, cumpliendo con los patrones arquitectónicos y de diseño requeridos, permitió obtener un módulo funcional para el sistema.
- La aplicación de pruebas funcionales de caja negra demostró el correcto funcionamiento de las funcionalidades asociadas al módulo garantizando la adaptabilidad y utilidad del mismo al sistema.
- Con el rediseño e implementación del módulo de Recurso Humanos se logró incluir de forma funcional la gestión de planes de formación en la herramienta XEDRO GESPRO.

Recomendaciones

Profundizar en la investigación al área de la formación, con el objetivo de mejorar y agregar funcionalidades al sistema atribuyendo al mismo de una eficiente gestión de Recursos Humanos.

Realizar una mejora en la optimización del algoritmo utilizado para proponer planes de formación de manera tal, que al ejecutar una cantidad moderada de usuarios y actividades de formación, disminuya el tiempo de respuesta.

Referencias

Hernández León, Rolando Alfredo. 2009. *Una Introducción a la Gestión de Proyectos.* 2009.

Juárez García , Norma y Herrera Arellano, Clara. 2008. Los Recursos Humanos. *Los Recursos Humanos.* [En línea] 2008. http://www.losrecursoshumanos.com/contenidos/1802-la-reingenieria.html.

Achell, Juan Felipe Pons. 2010. *Análisis teórico del PMBOK® y su puesta en práctica en proyectos de Edificación.* Universidad Politécnica de Valencia. Valencia : s.n., 2010. tesis de maestria.

Albaladejo, Xavier . 2009. *Scrum: un proceso de trabajo 2.0.* 2009.

Alfonso, Arlan Hernández. 2012. *Análisis y diseño del módulo de gestión de los recursos humanos para el sistema GESPRO.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2012. Tesis de grado.

Buschmann, F, y otros. 1996. *Pattern – Oriented Software Architecture. A System of Patterns.* s.l. : John Wiley & Sons, 1996.

Catalano, Ana, Avolio de Cols, Susana y Sladogna, Mónica. 2004. *Diseño curricular basado en normas de competencia laboral: conceptos y orientaciones metodológicas.* Buenos Aires : OIT/Cinterfor, 2004. ISBN: 987-1182-25-2.

Chiavenato, Idalberto . 2007. *Administración de Recursos Humanos El Capital Humano de las Organizaciones.* [trad.] Pilar Mascaró Sacristán. s.l. : McGraw-Hill/Interamericana, 2007. pág. 699. 9701061047.

Cuesta Santos, Armando. 2010. *Tecnología de Gestión de Recursos Humanos.* La Habana : s.n., 2010.

Cuza García, Betsy. 2013. *Algoritmo para la elaboración de planes de formación profesional basados en competencias laborales para proyectos desarrolladores de software.* GESPRO, Universidad de las Ciencias Informáticas. La Habana : s.n., 2013. Tesis de maestría.

Deitel, Harvey M. y Deite, Paul J. 2003. *Cómo programar en C++.* 2003.

Ecured. [En línea] http://www.ecured.cu/index.php/Lenguaje_de_Programaci%C3%B3n_Ruby#Caracter.C3.ADsticas_generales_del_lenguaje.

Fidel Gil, Javier Albrigo, Javier Do Rosario. 2005. *SISTEMAS DE GESTIÓN DE BASE DE DATOS SGBD / DBMS.* Facultad Experimental de Ciencias y Tecnología, Departamento de Computación, Universidad de Carabobo. Valencia, Carabobo : s.n., 2005.

Fleitas Triana, Sonia. 2000. *Modelando el proceso de gestión de recursos humanos.* Ciudad de la Habana : s.n., 2000.

Freeman, Peter. 1987. *" A Perspective on Reusability". IEEE Tutorial: SoftwareReusability.* Michigan : Computer Society Press of the IEEE, 1987. pág. 297. 0818607505.

Gamma, Erich, y otros. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : ADDISON-WESLEY, 1994.

Gartner. 2013. *Magic Quadrant for IT Project and Portfolio Management.* Stanford, Estados Unidos : Gartner Inc., 2013. 0471649848.

Gómez Mejía, Luis , Balkin, David y Cardy, Robert. 2008. *Gestión de recursos humanos.* [ed.] Pearson Educació. 5ta. s.l. : Pearson Prentice Hall, 2008. pág. 720. 848322402X.

Hammer, Michael y Stanton, Steven A. 1997. *La revolución de la reingeniería.* s.l. : Diaz de Santos, 1997. pág. 384. 9788479783099.

Hansson, David Heinemeier. Ruby on Rails. *Ruby on Rails.* [En línea] http://rubyonrails.org.

High Tech Tools Providers del Perú. 2012. http-peru Proveedores de Tecnologías de Información. *http-peru Proveedores de Tecnologías de Información.* [En línea] 2012. www.http-peru.com/postgresql.php.

Hitpass Heyl, Bernhard. 2011. *BPMN 2.0 Manual de Referencia y Guía Práctica.* s.l. : Editora Microbyte Ltda., 2011. pág. 269. 9563451821.

Hutchison, David. 2008. *High Confidence Software Reuse in Large Systems: 10th International Conference on Software Reuse, Proceedings.* Beijing, China : Springer, 2008, 2008. 9783540680734.

Kaisler, Stephen H. 2005. *Software Paradigms.* New Jersey : John Wiley & Sons, 2005. ISBN 0-471-48347-8.

Kniberg, Henrik. 2007. *Scrum and XP from the Trenches.* s.l. : Crisp, 2007. pág. 140.

Korgent Inc. 2008. *NetBeans 6 in Simple Steps si.* New Delh : Dreamtech Pres, 2008. ISBN 10-81-7722-897-8.

Laboratorios de Gestión de Proyectos. 2012. GESPRO 12.05. *GESPRO 12.05.* [En línea] Laboratorios de Gestión de Proyectos, 2012. [Citado el: 15 de octubre de 2013.] http://gespro-help.prod.uci.cu/.

Letelier, Patricio y Penadés, Maria del Carmen. 2006. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* España : Universidad Política de Valencia, 2006.

Manganelli, Raymond L. y Klein, Mark M. 1995. *Cómo hacer reingeniería.* Bogotá : Norma, 1995. pág. 349. 9580477523.

Martínez Vigil, Iván . 2012. *Módulo de Gestión de Riesgos versión 2.0 para el sistema GESPRO 12.05.* Universidad de las Ciencias Informáticas. La Habana,. Cuba : s.n., 2012. pág. 80, Tesis diploma.

Matsumoto, Yukihiro. 2001. *Ruby in a Nutshell.* s.l. : O'Reilly Media, 2001.

Mendes Calo, Karla, Estevez, Elsa y Fillottrani, Pablo. 2010. Journal of Computer Science & Technology. *Journal of Computer Science & Technology.* [En línea] junio de 2010. http://journal.info.unlp.edu.ar/journal/journal28/papers/JCST-Jun10-4.pdf..

Mesa, Jose Luis. 2012. *Patron de Acceso a datos (DAO).* 2012.

Muller, Hausi A., Norman, Ronald J. y Slonim, Jacob . 2011. *Computer Aided Software Engineering.* New York : Springer, 2011.

Netbeans. 2011. Bienvenido a NetBeans. [En línea] 2011. http://netbeans.org/index_es.html.

Object Management Group. 2014. Unified Modeling Language™ (UML®). *Unified Modeling Language™ (UML®).* [En línea] Object Management Group, Inc., 2014. http://www.uml.org/.

Palacio, Juan. 2006. NavegaPolis.net. *NavegaPolis.net.* [En línea] 2006. http://www.navegapolis.net/files/s/NST-003_01.pdf.

PCC. 2011. *Lineamientos de la Política Económica y Social del Partido y la Revolución.* Partido Comunista de Cuba. La Habana : s.n., 2011. pág. 41.

Pete Deemer, Gabrielle Benefield, Craig Larman, Bas Vodde. 2009. *The Scrum Primer.* s.l. : Scrum Training Institute, 2009.

PgAdmin III. 2012. PgAdmin3 PostgreSQL Tools. *PgAdmin3 PostgreSQL Tools.* [En línea] 2012. http://www.pgadmin.org/.

Piñero, Pedro. 2010. *Conferencia 1: Contratación, Factibilidad de Proyectos y Negociación.* GESPRO, Universidad de las Ciencias Informáticas. Ciudad de la Habana : s.n., 2010.

Piñero, Pedro y colectivo de autores. 2010. *GESPRO Paquete para la gestión de proyectos.* GESPRO, Universidad de las Ciencias Informaticas. Ciudad de la Habana : Universidad de las Ciencias Informaticas, 2010.

PMI. 2013. *Project Management Body of Knowledge.* Filadelfia : Project Management Institute, 2013. pág. 400. 9781933890517.

PostgresSQL. 2011. PostgresSQL. *PostgresSQL.* [En línea] 2011. http://www.postgresql.org/about/.

Pressman, Roger S. 2005. *Ingenieria de Software, Un enfoque práctico: Sexta Edición.* s.l. : McGraw Hill, 2005.

Ramos, Gastón. 2010. *Active Record, sabor Ruby.* 2010.

Rational Software. 2009. IBM. *IBM.* [En línea] 2009. http://www-03.ibm.com/software/products/es/rosemod.

Real Academia Española. 2001. Real Academia Española. *Real Academia Española.* [En línea] 2001. http://lema.rae.es/drae/?val=reutilizaci%C3%B3n.

Rodriguez Gutíerrez, Javier Jesus. 2007. Depertamento de Lenguajes y Sistemas Informaticos Universidad de Sevilla. *Depertamento de Lenguajes y Sistemas Informaticos Universidad de Sevilla.* [En línea] 2007. http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2007. *El Lengiaje Unificado de Modelado Manual de Referencia.* 2da Edicion. s.l. : Addison-Wesley Object Technology Series, 2007.

—. 2000. *El lenguaje Unificado de Modelado, Manual de Referencia.* s.l. : Addison Wesley, 2000. ISBN: 8478290370.

Salavert, Isidro Ramos y Lozano Pérez, María Dolores. 2000. *Ingeniería del software y bases de datos: tendencias actuales.* España : Ediciones de la universidad de Castilla-La Mancha, 2000. ISBN 84-8427-077-7.

Schwaber, Ken y Sutherland, Jeff . 2013. *La Guía de Scrum™. La Guía Definitiva de Scrum: Las Reglas del Juego.* Scrum.org. 2013. pág. 19.

Socas Alvez, Marisleydis y Rodríguez Verdecia, Lissette. 2007. *MIR-SWG: MODELO DE INGENIERÍA DE REQUISITOS PARA SOFTWARE DE GESTIÓN EN LA FACULTAD 3.* Universidad de Ciencias Informáticas. La Habana, Cuba : s.n., 2007. pág. 80, Tesis diploma.

Targetware. 2013. software.com.ar. *software.com.ar.* [En línea] 2013. http://www.software.com.ar/visual-paradigm-para-uml.html.

Torres Quiñones, Karina Mileisis . 2014. *Estrategia de formación integrada en Gestión de Proyectos Informáticos.* GESPRO, Universidad de las Ciencias Informáticas. La Habana : s.n., 2014. Tesis de maestría.

Valdivia Cerda, Anibal Sayid. 2003. *Modelo de diseño en UML e implantación del Sistema de Gestión para las dependencias de la universidad de Colimas.* Ingeniería Mecánica y Eléctrica, Universidad de Colima. Colima : s.n., 2003. Tesis de Maestria.

Valenciano, Jaime de Pablo. 2009. *La elaboración de un plan de formación e innovación.* Asturias, España : s.n., 2009.

Visual Paradigm. [En línea] [Citado el: 8 de octubre de 2013.] http://www.visual-paradigm.com/.

Visual Paradigm International. 2013. Visual Paradigm. *Visual Paradigm.* [En línea] 2013. http://www.visual-paradigm.com/.

Wilson, Leslie Blacket. 1993. *Comparative Programming Languages.* New York : Addison-Wesley, 1993. ISBN 0-201-56885-3.

Yiiframework. 2010. Working with databases. *Working with databases.* [En línea] 2010. [Citado el: 19 de Marzo de 2014.] http://www.yiiframework.com/doc/guide/1.1/es/database.ar.

Anexos

Anexo 1: Descripción de los Requisitos Funcionales

Ver la plantilla **0113_Especificación de Requisitos de Software** adjunta a esta investigación, sección referida a los Requisitos Funcionales.

Anexo 2: Casos de Prueba Basados en Requisitos

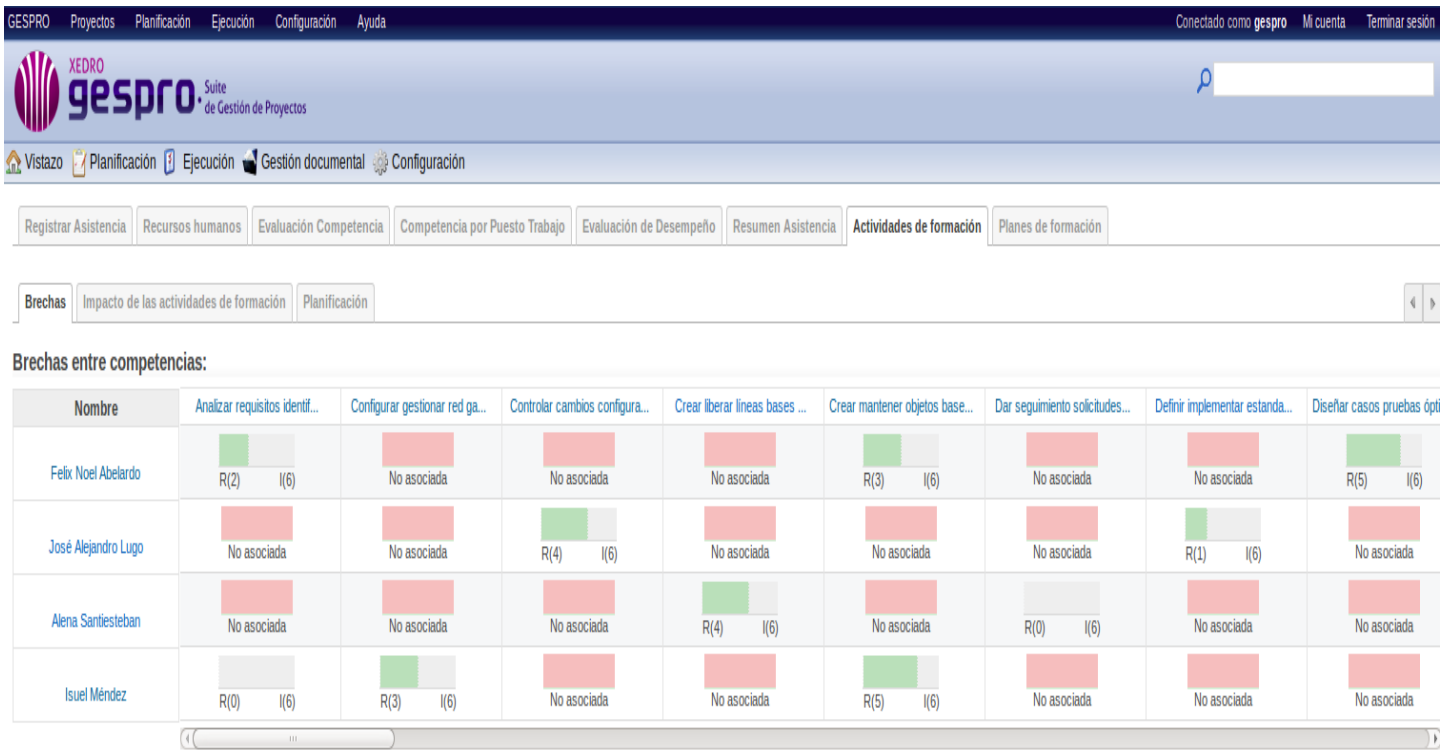
Ver la carpeta **Casos de prueba basados en requisitos** adjunta a esta investigación, en la cual se encuentran los diferentes casos de pruebas.

Anexo 3: Pruebas de Caja Negra

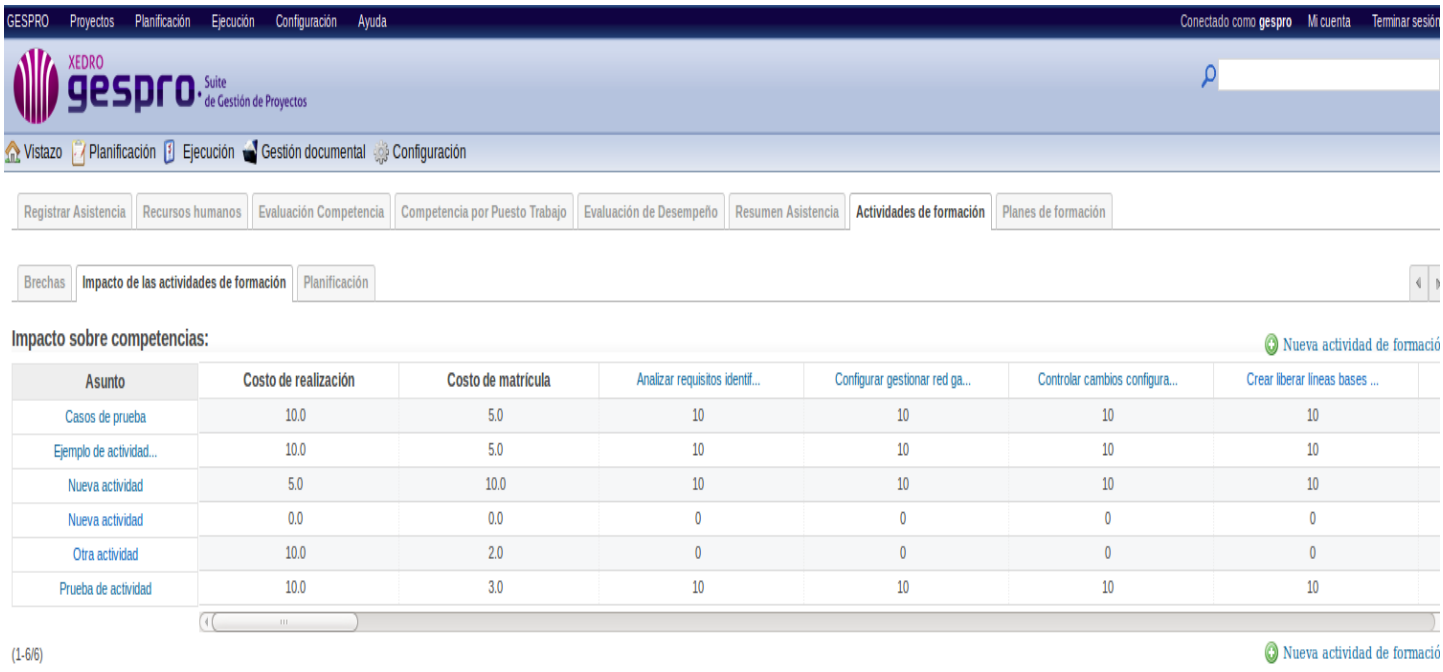
Ver la carpeta **Prueba de Caja Negra** adjunta a esta investigación, en la cual se encuentran las pruebas sometidas a las interfaces del módulo.

Anexo 4: Vistas de las interfaces relacionadas a las nuevas funcionalidades del módulo.

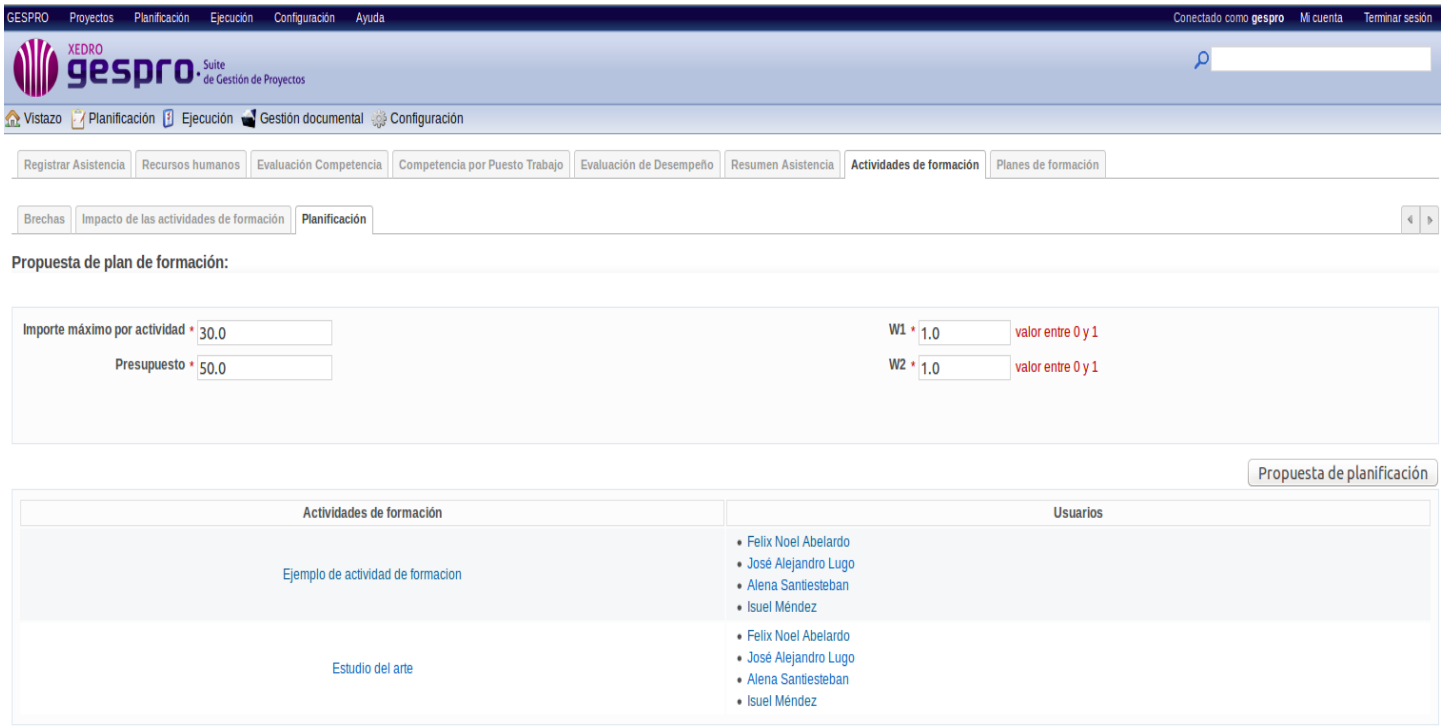
Vista de las brechas



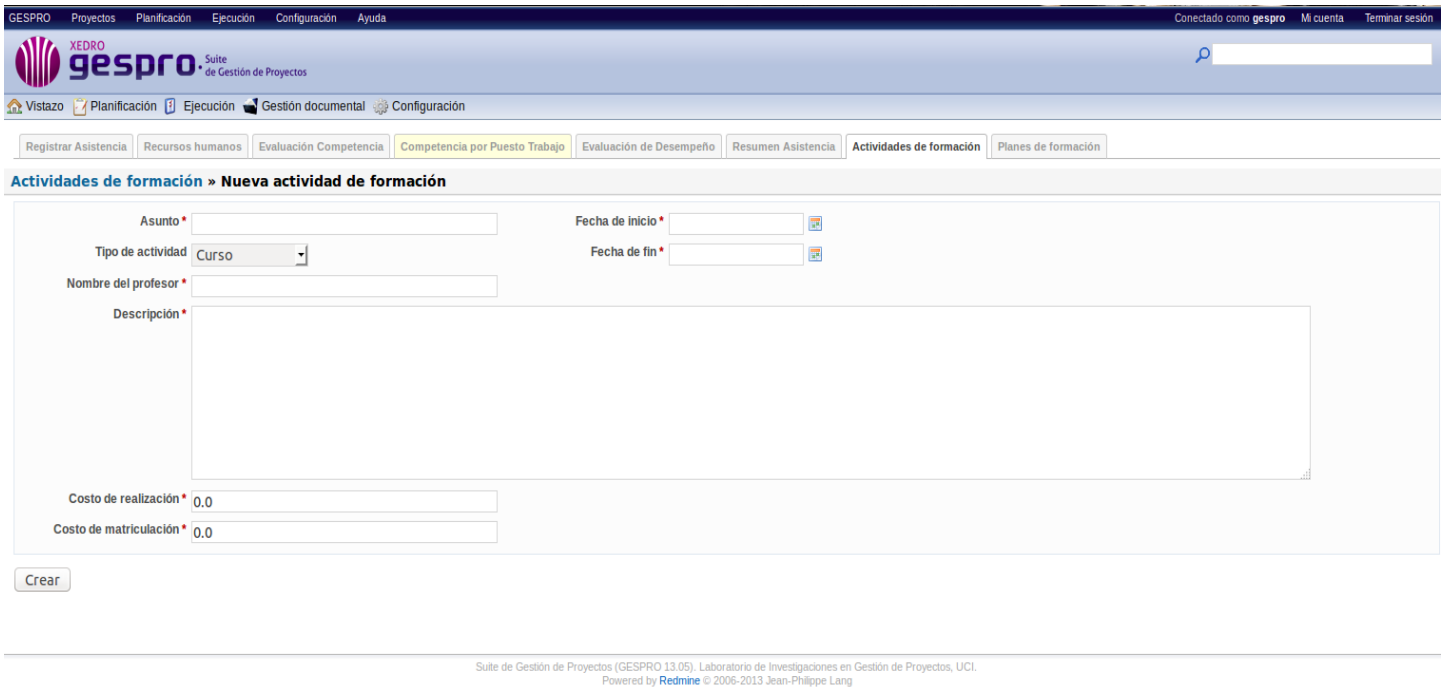
Vista de impacto de las actividades sobre las competencias



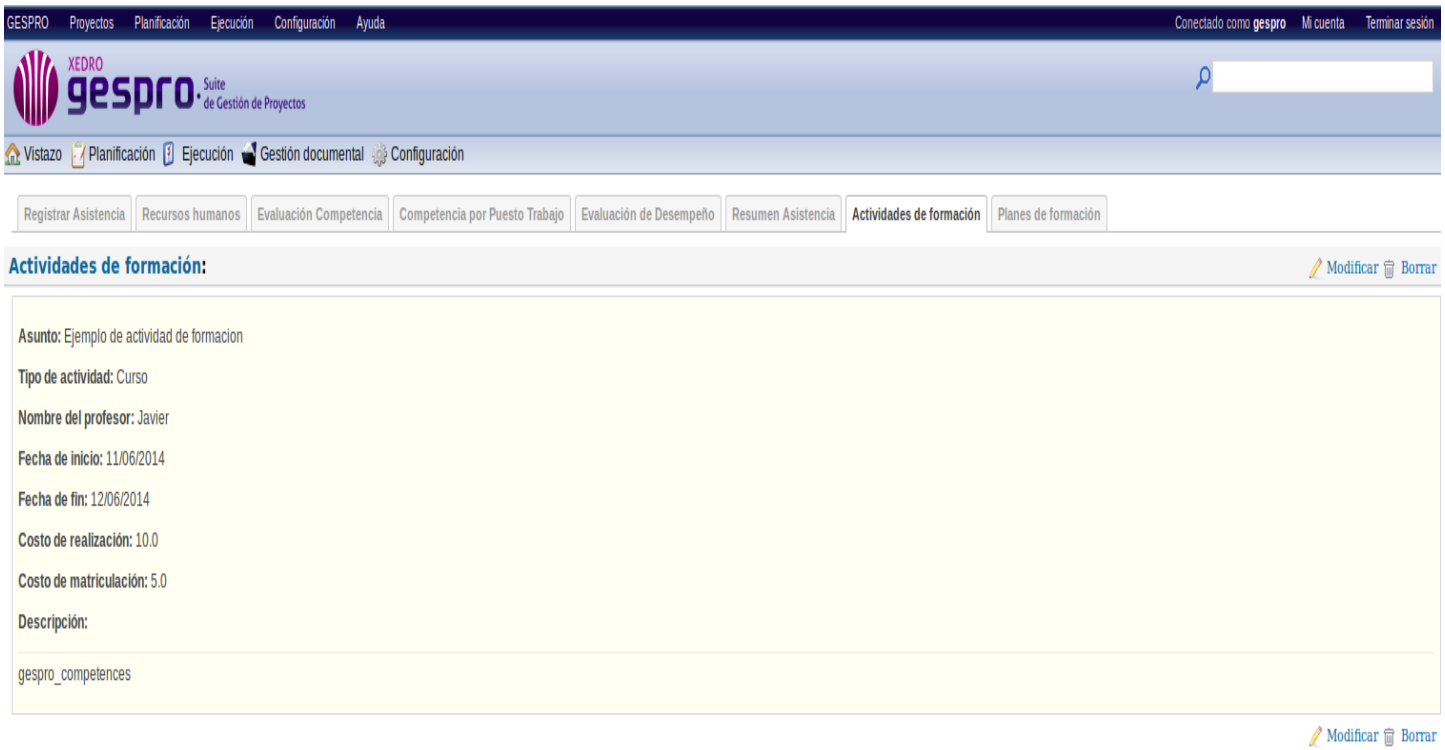
Vista para elaborar planes de formación



Vista para crear actividades de formación



Vista de la actividad de formación



Vista donde se muestran los planes de formación