Universidad de las Ciencias Informáticas

Facultad 3

Herramienta para la representación de las relaciones existentes en un proyecto de software a través de un modelo de grafos

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

Autores

Maiko Hernández Martínez

Rigoberto Peña Cabrera

Tutores

DrC. Rafael Rodríguez Puente

Ing. Eliober Cleger Despaigne

La Habana, Junio 2015

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores del presente trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Rigoberto Peña Cabrera	Maiko Hernández Martínez
_____	_____
Firma del autor	Firma del autor
Ing. Eliober Cleger Despaigne	DrC. Rafael Rodríguez Puente
_____	_____
Firma del tutor	Firma del tutor

AGRADECIMIENTOS

A mi madre, por su infinita entrega, dedicación y ser la responsable de mi formación como profesional. A mi padre, por su incondicionalidad y aceptación luego de 15 años sin su presencia. A mi familia por su entera confianza en mi superación. A mi novia, por su disposición y comprensión, obteniendo un lugar importante en mi vida. A mi compañero de tesis, que en el poco tiempo que compartimos, puedo llamarlo mi hermano. A nuestros tutores, por la entrega diaria al desarrollo de este trabajo. A las amistades de mi grupo por su entera disposición en los momentos difíciles, así como aquellas que siempre estuvieron cuando las necesité. Agradecer de manera general a todas las personas que hoy comparten el logro de uno de los objetivos de mi vida.

Rigoberto Peña Cabrera

Agradecer en primer lugar a mis padres dos personas muy especiales, sin las cuales, ninguna de las cosas que han ocurrido en mi vida habrían sido posibles y si hoy estoy aquí parado delante de ustedes se lo debo todo a ellos. Gracias por la dedicación, la paciencia, los consejos, la educación y el amor que me han transmitido durante todo el camino recorrido y el que falta por recorrer, para mí es un placer y un regocijo ser su hijo. A mis tíos y primos que siempre están pendientes de mí para lo que necesite. A mi compañero de tesis que a pesar de habernos conocido hace apenas un curso hemos podido lograr grandes cosas, entre ellas una buena amistad y cumplir todas las metas propuestas. A mi tutor y más que tutor amigo, del cual hemos podido disponer en todo momento, orientándonos siempre por el camino correcto. A mis compañeros que se mantuvieron siempre a mi lado sin importar la situación. Y de manera general a todos los que han contribuido en mi desarrollo como profesional.

Maiko Hernández Martínez

RESUMEN

La gestión de los recursos humanos en los sistemas de gestión de proyectos, se ha convertido en un área de principal atención para los especialistas del tema, los cuales, apoyados por el uso de los criterios de medidas, pueden realizar las actividades requeridas de forma eficiente. En el contexto de la Universidad de las Ciencias Informáticas, el sistema de gestión de proyectos Xedro-GESPRO utiliza una serie de indicadores para medir el avance de los proyectos gestionados. Entre ellos, el índice de rendimiento de recursos humanos tiene el objetivo de medir el desempeño de cada persona ante las tareas asignadas. La incorporación de nuevos proyectos, así como los mecanismos del modelo relacional empleados para consultar la información, afectan de manera significativa el tiempo requerido para calcular dicho indicador. La situación antes descrita provoca demoras en la obtención de información relevante para la toma de decisiones de la dirección general de producción. Se propone la implementación de una herramienta que permita estructurar los datos existentes en el sistema Xedro – GESPRO en un modelo orientado a grafos, que contenga los conceptos y relaciones requeridos para el cálculo de los indicadores de forma eficiente; así como obtener diferentes vistas sobre el grafo para el apoyo a la toma de decisiones en el ámbito de los recursos humanos.

Palabras claves: gestión de proyectos, grafo, indicadores, modelo, recursos humanos.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La gestión de proyectos de software (GPSW), como parte del proceso de desarrollo de aplicaciones informáticas, provee las técnicas y herramientas para el logro de las metas propuestas en una entidad productora de software. Una correcta gestión de proyectos no garantiza en su totalidad el éxito de los proyectos, por el contrario, una mala gestión usualmente incide en el fracaso del mismo (Sommerville, 2006).

Como parte del avance de las Tecnologías de la Información y las Comunicaciones, así como la informatización de la sociedad y sus campos científicos y productivos, la GPSW ha sido beneficiada con la creación de diferentes sistemas informáticos para la realización de sus actividades, definiéndose con el término de Sistemas de Gestión de Proyectos. Estos sistemas actualmente permiten la informatización de algunos de sus procesos como la planificación, la estimación de costos, la gestión de los recursos humanos, entre otros.

Desde el surgimiento de la Universidad de las Ciencias Informáticas (UCI), existieron diversas herramientas para la GPSW (López Zaldívar, 2012), las cuales fueron utilizadas hasta el año 2010, fecha en la que se desarrolla la herramienta integral para la gestión de proyectos: Xedro – GESPRO (Lugo García, 2012). Dentro de la actividad relacionada con los recursos humanos, el sistema analizado ha incorporado una serie de indicadores para el control de la ejecución de los proyectos, determinados mediante lógica borrosa (Lugo García, 2012). Uno de los indicadores utilizados en la gestión de los recursos humanos es el Índice de Rendimiento de los Recursos Humanos (IRRH), el cual muestra el estado o avance de los recursos humanos ante la realización de las tareas asignadas.

Este indicador es calculado en el sistema mediante cuatro sub-indicadores: Índice de rendimiento del Recurso Humano con respecto al Trabajo (IRHT), Índice de rendimiento del Recurso Humano con respecto a la Eficacia (IRHE), Índice de rendimiento del Recurso Humano con respecto a su Aprovechamiento (IRHA) e Índice de rendimiento de los Recursos Humanos con respecto a la Eficiencia (IRHF). Estos son determinados por diferentes mecanismos de consulta sobre los datos del sistema. Su estructuración se basa en el uso del modelo relacional, utilizando para ello el sistema gestor de base de datos (SGBD) PostgreSQL. En la etapa inicial de explotación del sistema Xedro-GESPRO, el contenido de datos registrados permitía el cálculo de estos indicadores sin inconveniente alguno. Un análisis posterior permitió identificar que "*los algoritmos implementados se enfrentaban a una subida de tiempo*

en su ejecución en tanto mayor fuera la cantidad de datos a procesar desde la plataforma" (Lugo García, 2012). Aunque fueron aplicadas medidas para disminuir estos factores, como el uso de técnicas para la optimización de bases de datos y en específico sus consultas, las mismas constituyen soluciones temporales.

Por otro lado, en un proyecto de desarrollo de software, se pueden identificar relaciones que se establecen entre los recursos humanos que se gestionan (personas) y los diferentes conceptos de la GPSW tales como hito, rol, tarea, proyecto, metodología, entre otros; y a su vez, entre los mismos recursos humanos. Dichas relaciones pueden ser modeladas a través de un modelo teórico conceptual presentado en (Rodríguez Puente & Ril Valentin, 2014).

Mediante el modelo propuesto, se pueden realizar diversos análisis relevantes para la toma de decisiones en el ámbito de los recursos humanos, como el balance de carga de tareas entre los integrantes del proyecto, propiciando información sobre qué personas tienen una sobrecarga y cuáles pueden asimilar una mayor carga de trabajo; la búsqueda de posibles líderes en los equipos de desarrollo; el cálculo del desempeño de las personas en diferentes roles, lo cual puede propiciar una eficaz asignación ante un nuevo proyecto y los roles requeridos; los análisis de intermediación, exponiendo aquellas personas que jugaron un papel decisivo en hitos importantes del proyecto, entre otros; basados todos en los diferentes análisis que se pueden realizar sobre el modelo matemático grafo.

A través de una entrevista en profundidad realizada a los administradores del sistema Xedro – GESPRO, se pudo constatar que su arquitectura de datos no facilita la obtención de la información para su representación en un modelo orientado a grafo, debido a que su estructuración está basada en el modelo relacional. Por lo anteriormente descrito, se plantea el siguiente problema.

Problema

¿Cómo obtener una representación de las relaciones existentes en proyectos de software utilizando un modelo orientado a grafo, a partir de la fuente de datos del sistema Xedro – GESPRO, de manera que contribuya a la realización de análisis de indicadores relacionados con los recursos humanos de forma eficiente?

Objeto de estudio

Modelos basados en grafos para la representación de los conceptos de la gestión de proyectos.

Objetivo general

Desarrollar un sistema informático capaz de obtener la representación del modelo matemático grafo de las relaciones existentes en los proyectos de software a partir de la información registrada en el sistema Xedro – GESPRO, de manera que contribuya a la realización de análisis de indicadores relacionados con los recursos humanos de forma eficiente.

Campo de acción

Cálculo de indicadores relacionados con la gestión de recursos humanos en proyectos de desarrollo de software.

Hipótesis

Si se desarrolla un sistema informático capaz de obtener la representación del modelo matemático grafo de las relaciones existentes en los proyectos de software a partir de la información registrada en el sistema Xedro – GESPRO, entonces se contribuirá al análisis de los indicadores relacionados con la gestión de los recursos humanos de forma eficiente.

Objetivos específicos

- Realizar el marco teórico-referencial de la investigación, de manera que contribuya a la comprensión de la base conceptual para la construcción del modelo, la identificación de soluciones similares y la definición de la metodología, herramientas y tecnologías para el desarrollo de la solución.
- Desarrollar un modelo basado en grafo, que integre los conceptos y relaciones necesarios para la implementación de los algoritmos utilizados en el cálculo del indicador IRRH, así como las vistas necesarias para el apoyo a la toma de decisiones.
- Desarrollar una herramienta informática como soporte al modelo orientado a grafo obtenido, que incorpore la transformación de los datos del sistema Xedro – GESPRO.
- Validar los algoritmos para el cálculo del indicador IRRH a través de un caso de estudio que permita realizar una comparación entre los propuestos en la presente investigación y los implementados en el sistema Xedro – GESPRO.

Tareas de la investigación

- Formalización de la base conceptual de la investigación, para la comprensión de la misma y el diseño de los algoritmos necesarios para dar solución al problema planteado.
- Realización de una búsqueda bibliográfica con el objetivo de identificar soluciones existentes y con ello definir posibles componentes a reutilizar.
- Fundamentación del mecanismo para hacer persistir información en forma de grafos.
- Realización de un análisis sobre los indicadores relacionados con los recursos humanos en el sistema Xedro – GESPRO, que permita definir los elementos necesarios para conformar el modelo orientado a grafo.
- Obtención del modelo orientado a grafo para la representación de los conceptos y relaciones asociados al cálculo del indicador IRRH.
- Definición de un conjunto de vistas sobre el modelo orientado a grafos que aporten información relevante para el apoyo a la toma de decisiones.
- Diseño e implementación de algoritmos necesarios para el cálculo de los sub-indicadores IRHT, IRHE, IRHA, IRHF.
- Realización de las tareas ingenieriles planteadas por la metodología seleccionada para el desarrollo de la herramienta GESPRO-Graph.
- Validación de los algoritmos para el cálculo de los indicadores de desempeño y rendimiento de recursos humanos a través de un caso de estudio y con ello realizar una comparación con los algoritmos implementados en el sistema Xedro – GESPRO.

Métodos de la investigación

Como vía para realizar el proceso científico investigativo con calidad, se utilizaron los siguientes métodos científicos:

- Histórico – lógico: es utilizado para el análisis del comportamiento evolutivo de las diferentes tecnologías comprendidas en la investigación científica como modelos de datos y sus tipologías, las tendencias actuales respecto a las bases de datos orientadas a grafos, las técnicas empleadas para la representación visual de grafos, determinando el grado de influencia sobre el trabajo que se presenta.

- Analítico – sintético: para el análisis de los diferentes contenidos y fuentes bibliográficas consultadas, haciendo procesos de síntesis sobre dicha información, arribando a conclusiones en la investigación.
- Sistémico: para la determinación de los componentes existentes en los sistemas analizados, en este caso, el sistema Xedro – GESPRO.
- Modelado: para la representación de las relaciones existentes entre las personas y los conceptos de la gestión de proyectos de software en el modelo propuesto, partiendo del modelo relacional. Es también utilizado en la confección de los diagramas del proceso de desarrollo de software, representando una abstracción de la solución.
- Revisión bibliográfica: para la elaboración de la base conceptual, la investigación de las herramientas analizadas en el estado del arte, los indicadores relacionados a la gestión de los recursos humanos, entre otros.
- Entrevista: para la identificación de los elementos en el modelo relacional utilizados para el cálculo de los indicadores en el sistema Xedro – GESPRO, a través de los especialistas que lo administran, adquiriendo mayor información para el proceso investigativo.

La presente investigación se estructura en tres capítulos que se describen a continuación.

Capítulo 1: Fundamentación teórica. En el capítulo se describen los principales conceptos que serán utilizados en la investigación. Se realiza un análisis a los indicadores utilizados para medir el rendimiento y desempeño de los recursos humanos utilizados en el sistema Xedro – GESPRO, así como los diferentes modelos basados en grafos y las herramientas utilizadas para el análisis y representación de grafos en el estado del arte. Se exponen las tecnologías y herramientas para conformar la solución y los criterios utilizados para su selección, así como la metodología de desarrollo a emplear.

Capítulo 2: Propuesta de solución. Se describe la solución propuesta ante el problema de investigación planteado, describiéndose el modelo orientado a grafos y los algoritmos para calcular los indicadores. Se expone el proceso ingenieril realizado mediante la metodología seleccionada, los elementos de la ingeniería utilizados como los patrones de diseño y la arquitectura empleada que permitirán el desarrollo de la herramienta.

Capítulo 3: Validación de la solución. Se verifican las funcionalidades del sistema desarrollado a través de la realización de pruebas de caja blanca y caja negra, describiéndose además un caso de estudio con el objetivo de verificar el cumplimiento de la hipótesis definida en el diseño metodológico.

Finalmente se presentan las conclusiones generales de la investigación y recomendaciones de los autores con vista a trabajos futuros.

CAPÍTULO 1: Fundamentación teórica

En el presente capítulo se introducen conceptos asociados al objeto de estudio, esclareciendo el significado de términos que serán usados durante el desarrollo de la investigación. En mayor medida se realizó un análisis sobre las diferentes herramientas que pueden dar solución al problema propuesto, conformando de esta manera el estado del arte, definiéndose una solución propia por parte de los autores. Se definió además la metodología de desarrollo de software, necesaria para guiar el desarrollo de la herramienta a confeccionar, incluyéndose las tecnologías, herramientas de desarrollo y métricas a utilizar para comprobar la calidad del proceso desarrollado.

1.1 Base conceptual

El modelo relacional es un modelo de datos surgido en 1969 por Edgar Frank Codd, en los laboratorios de la *International Business Machines*, orientado a la gestión de las bases de datos. Este se basa en el uso de tablas (compuestas de filas y columnas conformando una estructura matricial) para la representación de la información en forma de registros o tuplas, e incorpora como nuevo concepto la interpretación de una relación a través de una tabla formada por los elementos relacionales (Frank Codd, 1990). Este modelo se ha utilizado ampliamente en los SGBD. Dichos sistemas, forman una herramienta de apoyo para los administradores y diseñadores de bases de datos, permitiendo el almacenamiento, modificación y extracción de la información que se registra en las bases de datos que administran (D. Ullman & Widom, 1999). Para ello, cuentan con interfaces asequibles para los usuarios, dejando a un lado las antiguas consolas[1] para la realización de estas actividades.

A raíz del crecimiento de la información, y la búsqueda de una alternativa al modelo relacional para la representación de los datos, surge el concepto de bases de datos NoSQL, nombradas así por el término en inglés de "*Not only SQL*", o "No solo SQL" de su traducción al español. Este concepto no indica desechar el mencionado modelo relacional, como ha sido objeto de confusión, sino que lo abarca e incorpora nuevos conceptos en cuanto a estructuración y formas de consultar la información, dando solución a los problemas que surgen en los sistemas gestores de bases de datos relacionales en escenarios donde radican las redes y el alto crecimiento de información (Camacho, 2010). Las

[1] Aplicaciones del sistema operativo que trabajan directamente con capas de servicios y aplicaciones en segundo plano

principales características de estas nuevas formas de concebir una base de datos se basa específicamente en su alto rendimiento, su escalabilidad, replicación, y la desestructuración de los altos volúmenes de datos que gestionan (Almuñez Mandryka & Perez Santoyo, 2012).

Como aplicación específica al modelado de redes y su posterior análisis, una de las categorías de las bases de datos NoSQL ha cobrado mayor atención por parte de los especialistas: las bases de datos orientadas a grafos (Gracia del Busto & Yanes Enríquez, 2012). Estas representan su información en forma de nodos (o conceptos) y relaciones entre ellos, lo que posibilita la realización de diferentes análisis aplicando la teoría de grafos. La principal característica que aporta esta categoría es la conceptualización de los elementos de un grafo (díganse nodos y relaciones) como un dato en sí. Dada esta posibilidad, tanto en los nodos, como en las relaciones, pueden almacenarse diferentes atributos con sus respectivos valores (Robinson, Webber, & Eifrem, 2013).

Como sustento teórico de las bases de datos orientadas a grafos se encuentra el modelo matemático grafo, el cual expone las características algebro-matemáticas de este concepto. El mismo define en su composición un conjunto de vértices y aristas, los que representan los nodos y vértices del grafo respectivamente, así como una serie de teoremas para su comprensión y el análisis (Wasserman & Faust, 1994).

Para la solución de múltiples problemas, independientemente de su tamaño o importancia son utilizados como medio de solución los algoritmos. Un algoritmo según (H. Corman, E. Leiserson, L. Rivest, & Stein, 2009) es "*un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad*". Describe un proceso que parte de un estado inicial, puede contener o no entradas, sigue una serie de pasos lógicos (serie de instrucciones) hasta obtener un resultado o salida final.

La eficiencia de un algoritmo, en las ciencias de la computación, es el término utilizado para describir las propiedades de los algoritmos relacionadas a los recursos que utilizan. Existen dos maneras de estimar la eficiencia de un algoritmo: a través de su complejidad temporal y de la complejidad espacial.

La complejidad espacial se enfoca en los recursos utilizados durante su ejecución (Martinez Gil & Martín Quetglás, 2003). En términos informáticos, esta se basa en el uso de recursos del sistema informático como porciento y cantidad de memoria utilizada, cantidad de tareas realizadas por el microprocesador, así como tamaño en los medios de almacenamiento, entre otros aspectos. En cambio, la complejidad

temporal se basa en el tiempo requerido por el algoritmo para dar su resultado y teniendo en cuenta la problemática descrita se realiza un enfoque sobre la misma para la confección de la solución.

Teniendo en cuenta la necesidad de realizar una transferencia de información desde la fuente de datos del sistema Xedro – GESPRO, se introduce un conjunto de acciones pertenecientes a los procesos de extracción, transformación y carga (ETL). Dicho término es utilizado para referirse al proceso de mover y transformar la información de un sistema a otro (Grupo PowerData, 2013). Las fases o secuencias de un proceso ETL son las siguientes: extracción de los datos de una o varias fuentes, transformación de dicha información a través de técnicas de limpiezas, selección o reformateo, y por último la carga de la información en el sistema de destino.

La fuente de datos origen, como uno de los componentes del proceso ETL descrito, se caracteriza por ser un sistema donde la información sea almacenada o persistida en el tiempo. En la problemática descrita, el registro de toda la información relacionada a los recursos humanos de interés para este trabajo, se realiza en el sistema de gestión de proyectos Xedro – GESPRO. Un sistema de gestión de proyectos, es un término utilizado en la ingeniería de software referenciando un conjunto de herramientas automatizadas, para llevar a cabo las actividades que define la gestión de proyectos, tales como la planificación de proyectos, manejo y control de presupuesto, asignación de recursos, la colaboración y comunicación, el manejo de la calidad, la documentación y la administración de subsistemas relacionados (Project Management Institute, 2004). En la UCI, se utiliza hasta la fecha, el sistema de gestión de proyectos Xedro – GESPRO, en su versión 13.05, siendo transformada en varias ocasiones con el fin de optimizar sus actividades.

Relacionados con la gestión de proyectos, los indicadores dentro de una organización permiten el conocimiento y valoración de sus elementos claves, así como para medir su desempeño y resultados. Según (Project Management Institute, 2004) los indicadores están presentes en casi la totalidad de los procesos de la gestión de un proyecto, y son elaborados por los líderes, siendo parte del ciclo de vida de un proyecto. Su estimación dentro de una organización permitirá realizar comparaciones entre los resultados obtenidos y los esperados por los evaluadores.

1.2 Indicadores de gestión de recursos humanos

Como parte fundamental de la gestión de los recursos humanos en un proyecto de software, los indicadores constituyen una pieza fundamental para la medición del desempeño que logran en base del cumplimiento de los objetivos de la organización. Constituyen parte del conjunto de elementos que apoyan la toma de decisiones estratégicas en la dirección de un proyecto (Delgado Victore, 2003).

El sistema integrado de gestión de proyectos de la UCI, Xedro – GESPRO, utiliza actualmente el indicador IRRH, el cual es determinado mediante lógica difusa a través de cuatro sub-indicadores, definiendo rangos difusos para una serie de etiquetas nominales y para el proceso inverso en el que se obtienen los valores representativos de dichas etiquetas. Este indicador propone el análisis del desempeño del recurso humano en torno al desarrollo, impacto y prioridad de las tareas asignadas (Lugo García, 2012).

El sub-indicador IRHT estima la correlación[2] entre la estandarización de los tiempos estimados y los tiempos dedicados de las tareas relacionadas con el recurso humano. Los valores permisibles para este sub-indicador están en el intervalo de [-1; 1], siendo mejor el resultado para valores cercanos a uno. Se calcula según la ecuación (1):

$$IRHT = CORR\left(\left\{\frac{TE_{Tarea} - AVG(TE_{Tarea})}{STDV(TE_{Tarea})}\right\}, \left\{\frac{TD_{Tarea} - AVG(TD_{Tarea})}{STDV(TD_{Tarea})}\right\}\right) \quad (1)$$

Donde:

- TE_{Tarea}, TD_{Tarea}: tiempo estimado y dedicado de una tarea respectivamente.
- AVG y STDV: promedio y desviación estándar respectivamente.

El sub-indicador IRHE expresa la relación entre la cantidad de tareas cerradas evaluadas de Bien o Excelente y el total de tareas cerradas del proyecto relacionadas con el recurso humano en cuestión hasta una fecha de corte definida por el especialista que realiza el análisis. Adquiere valores en el intervalo [0; 1], siendo mejor el resultado para valores cercano a uno. Este indicador mide la calidad de

[2] Función estadística que determina el comportamiento de una variable con otra. En este caso es utilizada la correlación de Pearson.

las tareas que realiza el recurso humano, midiendo para ello las evaluaciones que obtiene, así como la importancia o prioridad de la misma. Se determina mediante la fórmula (2):

$$IRHE(dd/mm/aaaa) = \frac{CTBERH}{CTC} \quad (2)$$

Donde:

$$CTBERH = 3 * CTCPABE + 2 * CTCPNBE + CTCPBBE$$

$$CTC = 3 * CTCPA + 2 * CTCPN + CTCPB$$

- CTCPABE: cantidad de tareas cerradas con prioridad alta evaluadas de bien o excelente.
- CTCPNBE: cantidad de tareas cerradas con prioridad normal evaluadas de bien o excelente.
- CTCPBBE: cantidad de tareas cerradas con prioridad baja evaluadas de bien o excelente.
- CTCPA: cantidad de tareas cerradas con prioridad alta.
- CTCPN: cantidad de tareas cerradas con prioridad normal.
- CTCPB: cantidad de tareas cerradas con prioridad baja.

El sub-indicador IRHA, expresa la relación entre los totales de tiempo planificado y el tiempo del que dispone un recurso humano para la realización de una tarea, desde una fecha de corte planteada por el especialista que realiza el análisis de los indicadores. Su interpretación, define rangos para el valor del indicador, expresando si se encuentra subutilizado (IRHA < 0.75), aprovechado (0.75 ≤ IRHA ≤ 1) o sobrecargado (IRHA > 1). Se determina mediante la fórmula (3):

$$IRHA(dd/mm/aaaa) = \frac{TTP}{TTD} \quad (3)$$

Donde:

- TTP: tiempo total planificado de las tareas de un recurso humano.
- TTD: tiempo total disponible del que dispone un recurso humano.

El sub-indicador IRHF expone el estado de avance o progreso del recurso humano en la realización de las tareas que tiene asignadas. Su forma de determinación se expresa mediante la relación entre la sumatoria de los porcientos de ejecución real y la sumatoria de los porcientos de ejecución planificado

para cada una de sus tareas según el período de análisis. Para su determinación se utiliza la siguiente fórmula (4):

$$IRHF(dd/mm/aaaa) = \frac{\sum_{i=1}^{n} Ejec\_Real\_Tarea\_RH(i)}{\sum_{i=1}^{n} Ejec\_Planif\_Tarea\_RH(i)} \quad (4)$$

Donde:

- N: es la cantidad de tareas asignadas al recurso humano, desde el comienzo del proyecto hasta la fecha de corte.
- Ejec_Real_Tarea_RH (i): es el porciento de ejecución real de la tarea i.
- Ejec_Planif_Tarea_RH (i): es el porciento de ejecución planificada de la tarea i.

En la propuesta de solución, los indicadores antes mencionados sirven como base para la determinación de los elementos necesarios para confeccionar el modelo basado en grafo (los conceptos de nodos y aristas), así como para recrear los nuevos algoritmos basándose en el modelo propuesto.

1.3 Estudio del estado del arte

El uso de modelos basados en grafos como vía de solución es ampliamente utilizado en múltiples ramas de los campos científicos. En (Gyssens, Paredaens, Van den Bussche, & Van Gucht, 1994) se expone el uso de un modelo de bases de datos orientado a grafos, en el cual se utiliza como esquema y estructuración de los objetos el modelo matemático grafo. Para la manipulación de la información se utilizan cuatro operaciones básicas usando los principios de transformación en un grafo:

- Adición de un nodo.
- Adición de una arista.
- Eliminación de un nodo.
- Eliminación de una arista.

Como parte del desarrollo de software, y específicamente en la etapa de diseño, es utilizado para la elaboración de diagramas UML (Fischer, Niere, Torunski, & Zündorf, 2000). La mayoría de los diagramas elaborados a través de este lenguaje de modelado, utilizan el modelo grafo para su representación. Cercano al área de la gestión de proyectos, se expone en (Horowitz, 1967) el uso de grafos para el

cálculo del camino crítico de los hitos y el costo mínimo de un proyecto. Para realizar tal actividad, se hace uso de los algoritmos clásicos sobre grafos: camino de mínimo costo y *Dijkstra*[3] respectivamente. Aunque abarca el tema de la gestión de proyectos, analiza sus conceptos de forma separada: análisis de tiempo con grafos PERT para la determinación del tiempo de duración de un proyecto, análisis de costo de recursos humanos con grafos de uso de personal en las diferentes etapas o hitos del proyecto, entre otros. Si bien utiliza el modelo matemático grafo, este se encuentra limitado y sin hacer uso de un modelo específico y planteado formalmente por los autores.

Por otro lado, en (Rodríguez Puente & Ril Valentin, 2014) se expone un modelo teórico conceptual basado en grafo para la representación de las relaciones que existen en un equipo de desarrollo de software. Se plantea el uso de nodos o vértices para los conceptos de proyecto, persona, tarea, hito, rol, entre otros, así como las relaciones que se establecen entre ellos: un proyecto tiene hitos, en los hitos se planifican tareas, las tareas son asignadas a personas, entre otras (ver Anexo 1).

Aunque este modelo recoge los principales conceptos de la gestión de proyectos, representa una abstracción que puede ser condicionada para escenarios más específicos. Citando a sus autores: "*este modelo es posible adicionarle otros conceptos y relaciones, en dependencia de la necesidad de realizar análisis particulares, por lo que podemos decir que el mismo puede ser extendido para aplicarse en un ambiente específico*". Por tal motivo, será utilizado como base para la propuesta de solución al cumplir con las siguientes características:

- Es flexible ante el cambio, pues puede ser extendido para situaciones específicas, considerando el contexto de gestión de proyectos a utilizar.
- En el sistema de Xedro – GESPRO, tras un análisis de los conceptos de la gestión de proyectos que maneja, son identificados gran parte de los elementos que expone este modelo.

Como parte del objetivo de obtener una representación del modelo orientado a grafos, se realizó un análisis a múltiples herramientas de análisis de grafos. Dadas sus características y objetivos, las mismas pueden contribuir a la solución del problema planteado en la presente investigación.

Cytoscape es una herramienta de código abierto para la visualización de grafos, desarrollada en julio de 2002 soportada por la *National Resource for Network Biology*, y es utilizada ampliamente en el sector

[3] Algoritmo utilizado en la teoría de grafos para el cálculo del camino más corto entre un vértice y los restantes.

de la bioinformática para la representación de compuestos biológicos en redes, aunque en los últimos años han comenzado a realizar avances en otros espacios como Análisis de Redes Sociales y la Red Semántica. Soporta múltiples formatos de entrada de datos, destacándose el lenguaje de modelado de grafos (GML por sus siglas en inglés), así como SIF (*Simple Interaction Format*), XGMML, BioPAX, PSI-MI, GraphML, KGML (KEGG XML), SBML, OBO, *Gene Association* e incluso las hojas de cálculo de Microsoft Excel. Para la visualización de resultados permite la generación de formatos de salida como PDF, JPEG, PNG, SVG, BMP y PS (Shannon, y otros, 2003).

Una de las garantías que ofrece esta herramienta es la contextualización de la representación del grafo a visualizar en dependencia del usuario, representando tanto proteínas y genes del ámbito de la bioinformática, como otras entidades, de ahí que es posible detallar otro contexto de análisis. Cuenta con una amplia gama de aplicaciones dedicadas a temas específicos dentro de los campos antes expuestos, disponibles en su sitio oficial. En cuanto a servicios web, cuenta con aplicaciones para el consumo de bases de datos orientadas a grafos: Pathway Commons, IntAct, BioMart, y NCBI Entrez Gene, todas del campo bioinformático (Cytoscape, 2014). Como posible solución al problema planteado en la presente investigación, no cumple con los requerimientos básicos, pues no permite la importación de datos desde bases de datos relacionales, ni su transformación hacia una base de datos orientada a grafos.

Gephi es una herramienta interactiva para la exploración y visualización de redes basadas en grafos jerárquicos. Fue desarrollada por el consorcio *Gephi.org.* Es una herramienta de código abierto y libre, se utilizó para su concepción el lenguaje de programación Java, por lo cual es multiplataforma. Sus principales aplicaciones son el análisis exploratorio de los datos y las relaciones; las redes sociales y bioinformáticas, y la creación de imágenes enriquecidas para presentaciones. Contiene los análisis de métricas en el grafo como son la centralidad, cercanía, intermediación, coeficiente de *clustering*[4], HITS[5], modularidad, entre otros (Gephi.org, 2014). Para la visualización de los grafos, utiliza un motor de procesamiento visual en OpenGL[6], que conlleva a la visualización en tiempo real de 50000 hasta 1 millón de nodos sin crear sobre cargas al sistema, cargando previamente los nodos en memoria. Los estilos de visualización son ajustables, contando además con algoritmos para la estructuración del grafo como

[4] Agrupar en colecciones de datos más pequeños a la fuente inicial. Separar o categorizar por atributos comunes.
[5] Algoritmo utilizado para valorar y clasificar la importancia de una página web.
[6] Tecnología de gráficos utilizada en la informática para el pre-procesamiento de la información visual.

son el *ForceAtlas* y el *YiFanHu* multinivel. Cuenta con la visualización de los datos en forma de registros, facilitando intercambiar modelos para la consulta de la información, de grafos a tablas y viceversa (Gephi.org, 2014).

Esta herramienta permite importar desde bases de datos relacionales, contando con diferentes lenguajes como: MySQL, PostgreSQL, SQLite, entre otros, los cuales pueden ser actualizados mediantes complementos desde su sitio oficial. Como restricción de esta funcionalidad se encuentra la importación de los datos que conforman el grafo resultante. Esto limita a que exista un esquema relacional previamente definido, donde se referencien explícitamente los elementos del grafo, y en el sistema analizado (Xedro – GESPRO), esta información se encuentra dispersa en diferentes tablas. Se ha desarrollado de igual forma un complemento para la importación y exportación de datos sobre el sistema de bases de datos orientadas a grafos Neo4j, pero actualmente presenta incompatibilidades referentes a las versiones del sistema antes mencionado, y errores no corregidos en la conversión de algunos tipos de datos. Aunque esta herramienta posee múltiples beneficios para la reutilización de parte de sus componentes y funcionalidades, no se utiliza como parte de la propuesta de solución, ya que no se cuenta con una versión estable del complemento que facilita la importación y exportación.

Luego del análisis realizado, se arriba a la conclusión de que ninguna de las herramientas anteriores será utilizada como solución al problema planteado. Por tal motivo, se determina el desarrollo de una nueva propuesta empleando para ello algunos elementos importantes de las herramientas antes analizadas, como son:

- La infraestructura de carga de los datos del grafo en memoria (caché de datos) utilizada por Gephi, lo cual optimiza la búsqueda de nodos y relaciones en tiempo real, influyendo en el factor temporal.
- Los algoritmos de visualización de un grafo existentes en Gephi, para la organización de los elementos del grafo en las imágenes.
- El proceso de exportación de imágenes utilizado por Cytoscape para la generación de vistas en el grafo.

1.4 Metodología de desarrollo de software

Un elemento imprescindible en la conducción del proceso de desarrollo de software, es sin dudas, el uso de metodologías. Las mismas se clasifican en dos categorías principales: tradicionales y ágiles, lo que conlleva al análisis sobre cuál de ellas emplear en el presente trabajo. Varios han sido los debates sobre cuál es más efectiva en un cierto escenario, llegándose a un consenso de que es justamente el contexto del problema que se enfrenta, el que motiva a los desarrolladores y miembros de un equipo de desarrollo de software a tomar partido por una.

Dentro de las metodologías ágiles analizadas en (Tinoco Gómez, Pedro Pablo, & Salas Bacalla, 2010) se determina el uso de la metodología Programación Extrema (XP). Las condiciones del contexto que incidieron en su selección se enuncian a continuación:

- Revisiones periódicas con el cliente para definir cada una de las iteraciones necesarias, para corregir los errores identificados en cada uno de los entregables.
- La herramienta a desarrollar es de mediano tamaño.
- Necesidad de entrega rápida de la herramienta.
- Solución dependiente de un proceso investigativo.
- Reducida cantidad de miembros en el equipo de desarrollo (dos miembros).

XP es una metodología de desarrollo de software y dirección de proyectos, utilizada mayormente en proyectos de corta duración. Se basa en la simplicidad, la comunicación y la reutilización del código desarrollado. Pretende minimizar la complejidad de un proyecto y se enfoca directamente hacia el objetivo, hace uso de las relaciones interpersonales y la rapidez de reacción. El ciclo de vida ideal de XP consta de 6 fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte del proyecto (Beck & Andres, Extreme Programming Explained: Embrace Change, 1999).

El objetivo que se perseguía en el momento de crear esta metodología era la búsqueda de un método que hiciera que los desarrollos fueran más sencillos (Tinoco Gómez, Pedro Pablo, & Salas Bacalla, 2010). Entre las principales prácticas de XP se destacan:

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que se adelante en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.

- Refactorización: se basa en reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad, siendo más flexible al cambio. Las pruebas unitarias corroboran que la refactorización no introduzca ningún cambio.
- Programación en pares: consiste en que dos desarrolladores que participan en un proyecto compartan una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

Un aspecto esencial que se tiene en cuenta en el presente trabajo es la tolerancia al cambio, pues la modificación de un requisito visto por el cliente, trae consigo un tiempo adicional a la duración del proyecto. Por ello, se reafirma la selección de XP mediante la afirmación de su creador: "*Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar"* (Beck & Andres, Extreme Programming Explained: Embrace Change, 1999).

1.5 Herramientas y tecnologías

En cuanto a las herramientas y tecnologías para el desarrollo de la solución, se realiza un análisis según las siguientes finalidades: para la obtención y transformación del modelo matemático grafo, a partir de la fuente de datos del sistema Xedro – GESPRO, y para la visualización de dicho modelo. Para su selección, se tuvo en cuenta el criterio de soberanía tecnológica.

Para lograr la correcta representación del grafo en sus diferentes vistas, y proporcionar mayor claridad sobre las relaciones que se establecen entre sus elementos, se utilizó el marco de trabajo *Java Universal Network Graph Framework* (JUNG). Se define como una biblioteca de software que proporciona un lenguaje común y extensible para el modelado, análisis y visualización de datos que se puede representar en forma de grafo. Está escrito en Java y está diseñado para soportar una variedad de representaciones de entidades y sus relaciones, incluye implementaciones de una serie de algoritmos de la teoría de grafos, minería de datos y análisis de redes sociales, tales como rutinas para la agrupación, la descomposición, la optimización, la generación de grafos aleatorios, análisis estadístico, y el cálculo de las distancias de redes (SourceForge, 2003). En el presente trabajo se utiliza para la representación del grafo propuesto en sus diferentes vistas, con el objetivo de proporcionar mayor

claridad en cuanto a las relaciones que se establecen e identificar indicadores no estadísticos como la aparición de líderes de proyectos, personas con mayor o menor carga de trabajo, entre otros, que son identificados con mayor facilidad sobre la visualización del modelo propuesto en grafo.

Con el objetivo de realizar el cálculo de funciones estadísticas descritas en uno de los indicadores analizados (la correlación de Pearson y la desviación estándar en el indicador IRHT), se utiliza la biblioteca común para los desarrolladores en la rama matemática denominada *Apache Common Math*. La misma dispone de un conjunto de componentes matemáticos y estadísticos desarrollado por la fundación Apache™, con el objetivo de suplir las necesidades de los desarrolladores en el uso de algoritmos de las ciencias matemáticas y estadísticas. Todos los algoritmos que contiene se encuentran documentados, donde la mayoría han sido conformados con el uso de buenas prácticas y patrones de diseño referenciados (Apache.org, 2015). A diferencia de múltiples bibliotecas pre-configuradas, tiene como ventaja la integración directa y sencilla con los entornos de desarrollos de Java (depende solo de su importación). Tiene diversas aplicaciones en diferentes áreas del conocimiento como la inteligencia artificial, conteniendo los algoritmos de *clustering* como el *k-means,* algoritmos genéticos, filtrados de datos, entre otros; en matemáticas aplicadas en el manejo de métodos lineales, interpolación, diferenciación, números complejos, derivación, y problemas de optimización; así como diversas funcionalidades en la estadística, probabilidad y álgebra lineal.

Como lenguaje de desarrollo para confeccionar la solución, se define el uso de Java, un lenguaje de programación orientado a objetos, desarrollado por *Sun Microsystems* a principios de los años 90, y que actualmente forma parte de la compañía *Oracle*. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Con respecto a la memoria, su gestión no es un problema ya que ésta es gestionada por el propio lenguaje y no por el programador. Las aplicaciones Java están típicamente compiladas en un *bytecode*[7], aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador java también es posible (Oracle Corporation, 2015).

[7] Código intermedio entre el lenguaje de programación y el código binario de máquina.

En cuanto al código generado, este puede ser documentado directamente desde el propio lenguaje, utilizándose para ello etiquetas para las diferentes secciones, atributos, funcionalidades y sus parámetros, facilitando la comprensión para el desarrollador. Un paradigma de la programación que emplea es el orientado a objetos, conteniendo características que implica dicho paradigma como el uso de objetos, clases y herencia de clases (Llobet Azpitarte, y otros, 2008). En el desarrollo de proyectos de gran envergadura, brinda el soporte de múltiples interfaces de programación de aplicaciones (conocidas como API de sus siglas en inglés), permitiendo la incorporación de servicios de múltiples componentes. Esta característica facilita la integración de las restantes bibliotecas seleccionadas para la conformación de la solución, pues al ser desarrolladas en Java, pueden ser integradas de manera directa sin requerir configuraciones adicionales (Neo4j, 2010).

Como herramienta de desarrollo se seleccionó el entorno de desarrollo integrado (de la traducción del inglés *Integrated Development Environment* o IDE) Netbeans 8.0. Se caracteriza por ser libre y gratuito sin restricciones de uso, multiplataforma, considerado el IDE oficial del lenguaje de programación Java. Existe un número importante de módulos para extenderlo, es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun Microsystems fundó el proyecto de código abierto Netbeans en junio de 2000. La plataforma 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 módulos (Oracle Corporation, 2000).

Proporciona muchas garantías a los desarrolladores en la realización de aplicaciones como el *debugging*[8], las pruebas unitarias, el control de versiones mediante terceros (Git, Mercurial, Subversion), módulos y bibliotecas para el trabajo con otras variadas plataformas como trabajo con gráficos (OpenGL), servicios web, aplicaciones web tanto en Java como PHP y fuentes de datos de múltiples orígenes: bases de datos relacionales como PostgreSQL, MySQL, Oracle, SQL Server, entre otros. En el presente trabajo se utilizó para el desarrollo de los diferentes componentes que dieron solución al problema a resolver, así como para la realización de pruebas unitarias a las diferentes secciones que la requieran y el control de versiones del código fuente resultante.

[8] Proceso para identificar errores en algoritmos en tiempo de ejecución.

Para el diseño de la arquitectura a utilizar y sus principales componentes fue necesario el uso del Lenguaje de Modelado Unificado (UML) en su versión 2.0. Es utilizado fundamentalmente para el desarrollo de proyectos de software para diseñar arquitecturas de sistemas, los procesos, interfaces, componentes, flujos de actividades, así como los diferentes elementos que pueden modelarse de un sistema (Larman, UML y Patrones, 2004). Es empleado por la ingeniería de software en la mayoría de sus ramas, dada la gran vinculación con los procesos que la definen. UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema, ofreciendo un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados (Fowler & Sccott, 1999).

Para el desarrollo de los diferentes diagramas pertenecientes al diseño de la solución, fue empleada la herramienta de Ingeniería de Software Asistida por Computador Visual Paradigm en su versión 8.0. Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Utiliza como lenguaje de modelado de sus variados diagramas el lenguaje de modelado UML. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: *Enterprise, Professional, Community, Standard, Modeler* y *Personal* (Visual Paradigm International, 2004).

La herramienta está diseñada para una amplia gama de usuarios, incluidos los ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas, o para cualquier persona que esté interesada en la construcción de sistemas de software a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Además, Visual Paradigm soporta los últimos estándares de la notación UML en su versión 2.0. Tiene integrado la generación de código fuente y su documentación a partir de los diagramas concebidos, facilitando el desarrollo de aplicaciones desde su propio diseño. Además de generar, puede realizar ingeniería inversa sobre sistemas de bases de datos, generando los diagramas de entidad relación correspondientes a los modelos descritos en dichos sistemas. Genera diversos informes, imágenes y archivos descriptores de los diagramas que soporta,

utilizando formatos enriquecidos como PNG, JPEG y TIF para las imágenes, y XML, HTML y CORBA para la estructuración.

Uno de los aspectos fundamentales del presente trabajo, en cuanto a tecnología, es el uso de una herramienta para la gestión de un modelo orientado a grafo. Para ello, fue seleccionada la base de datos Neo4j en su versión 2.2. Neo4j es una base de datos NoSQL especializada fundamentalmente en el trabajo con grafos. Fue creada por la organización Neo Technology, la que actualmente proporciona múltiples servicios orientados al trabajo con dicho producto. Es libre, de código abierto, y tiene una amplia comunidad y aceptación, tanto por sectores académicos como empresariales. Se encuentra desarrollada en Java, y actualmente cuenta con múltiples complementos para el trabajo con otros lenguajes como .NET, PHP, Javascript, Ruby, Pyton y otros que continúan desarrollando por múltiples colaboradores. Se caracteriza por ser multiplataforma, existiendo diferentes versiones para los sistemas de Windows, Linux y Macintosh, y las arquitecturas de procesadores de 32 bits y 64 bits. Actualmente se encuentra en su versión 2.2, aunque están disponibles anteriores. Como parte de los beneficios que brinda en cuanto a la estructuración de la información, utiliza el modelo de grafo de propiedad, el cual expone el uso de atributos tanto en los nodos como en las aristas del grafo, optimizando el análisis y la consulta de información. Sus fuentes contienen múltiples recursos que agilizan la creación de productos para el desarrollador, como la incorporación de patrones de diseño, así como el enfoque orientado a objetos, permitiendo mayor reutilización y optimización (Van Bruggen, 2014).

Sus modos de usos son básicamente dos: como servidor, o como motor de almacenamiento embebido. La primera vía depende de un sistema para su alojamiento, donde se almacenan grandes cúmulos de información en el orden de gigabytes, la cual puede ser accedida y manipulada dado el uso deseado. Puede ser gestionada de forma distribuida mediante aplicaciones web, o consumir como servicios, permitiendo utilizar terceras herramientas para la consulta de su información y posterior procesamiento. Como segunda variante, se utiliza de forma embebida en pequeñas aplicaciones, comúnmente para el aprendizaje, donde suele utilizarse como motor de almacenamiento de información, sustituyendo a las comunes bases de datos relacionales. Esta variante tiene un amplio uso en el lenguaje de desarrollo Java, donde tiene fuertes referencias y documentación, e incluso patrocinado por sus creadores.

Es seleccionado como herramienta en el presente trabajo para la persistencia de la información en forma de grafo, utilizando para ello los recursos que brinda como los algoritmos de búsqueda y múltiples

patrones para la definición de estructuras orientadas a objetos, como los conceptos identificados en el modelo relacional y el orientado a grafo.

Definida como base de datos orientada a grafos Neo4j, y teniendo en cuenta las múltiples formas de consultar sus datos, se hace preciso seleccionar un lenguaje de consulta sobre el mismo, con el objetivo de lograr mayor rendimiento y menor consumo de recursos sobre el sistema a desarrollar. En (Holzschuher & Peinl, 2013) se realiza un análisis sobre las siguientes formas de consultar la información en base de datos orientadas a grafos de Neo4j:

- Neo4j embebido: se utiliza a través del marco de trabajo de Neo4j (conjunto de bibliotecas en Java), proporcionando un marco de trabajo flexible. Permite manejar los nodos de la base de datos embebida, así como con sus relaciones de forma directa.
- Neo4j REST: marco de trabajo utilizado para servidores de datos de Neo4j (uso como servidor) y el acceso a la información se realiza mediante servicios web.
- Neo4j embebido con Cypher: se utiliza de forma embebida, pero el acceso a la información se realiza mediante el lenguaje de consulta Cypher.
- Neo4j Cypher REST: utilizado en la forma de uso de servidor, el acceso a datos se realiza mediante consultas Cypher.
- Neo4j Gremlin REST: utilizada en la forma de uso de servidor, donde el acceso a datos se realiza mediante el lenguaje de consulta Gremlin. Este tiene mayor uso en el trabajo con caminos en un grafo.

Para la realización de las pruebas con dichas formas de consulta de la información, fueron empleadas las siguientes herramientas y tecnologías: como base de datos orientada a grafo Neo4j en su versión 1.8, para el consumo de servicios web el servidor *Apache Shindig* 2.5 beta y *Gremlin* del proyecto *Tinkerpop*. Los resultados arrojan que el uso del marco de trabajo embebido mediante el conjunto de bibliotecas de Java, proporciona un mayor rendimiento en cuanto a tiempo. En la Figura 1, se describe el comportamiento del tiempo de respuesta por cada tipo de petición realizada a la fuente de datos. A mayor complejidad de la consulta realizada (dígase buscar nodos relacionados entre sí, y crear una cadena sucesiva de relaciones), el tiempo empleado por cada marco de trabajo se afecta considerablemente para aquellos que emplean servicios web. Los categorizados en la forma de uso

embebida, tienen un mayor rendimiento en este aspecto, destacándose por sobre todos, el primero de los enunciados.

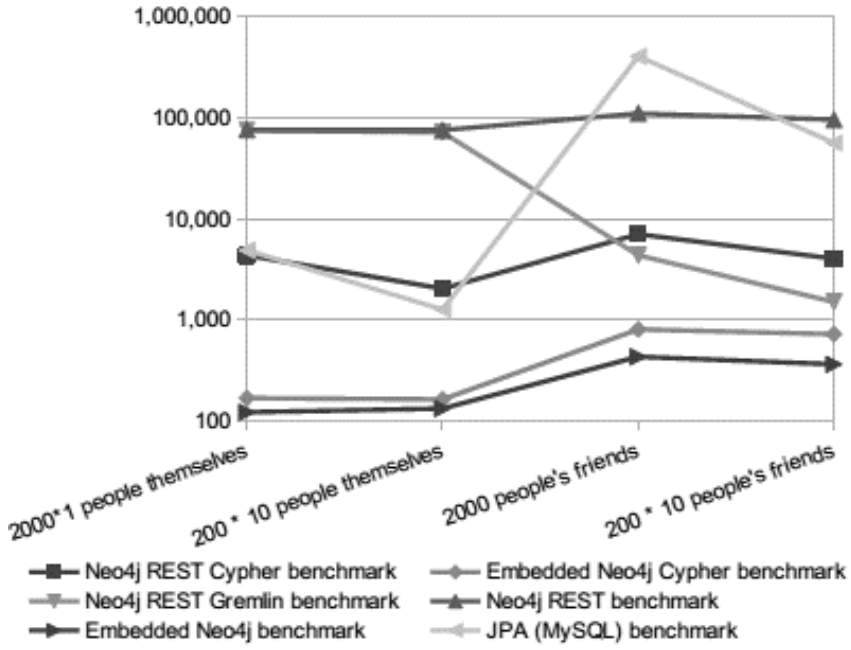

Figura 1 Resultados arrojados de las diferentes formas de consulta de información

Fuente: (Holzschuher & Peinl, 2013)

Por los resultados anteriores, se decide utilizar la forma de acceso embebida (*Embedded Neo4j benchmark* en la Figura 1) en la base de datos a conformar, teniendo como principal criterio de selección la búsqueda de la eficiencia en cuanto al costo temporal de dicha actividad.

Conclusiones parciales

En el presente capítulo se ha definido el marco teórico para la comprensión en detalle de la problemática a resolver. El análisis realizado a los indicadores de gestión de recursos humanos del sistema Xedro – GESPRO, posibilitó identificar los elementos del modelo relacional necesarios para conformar el modelo basado en grafo. En cuanto a las herramientas analizadas en el estado del arte, se pudo corroborar que ninguna puede ser utilizada como base para el desarrollo de la solución, definiéndose la creación de una propia por parte de los autores. Se han escogido para ello, un conjunto de herramientas y tecnologías unificadas por un elemento en común, el lenguaje de desarrollo Java.

CAPÍTULO 2: Propuesta de solución

En el presente capítulo se describe la propuesta de solución al problema planteado, así como los distintos elementos de la planificación, diseño e implementación relacionados con la metodología utilizada. Para ello, se inicia con el planteamiento del modelo basado en grafo para la estructuración de los datos, y los algoritmos para calcular los indicadores analizados en la sección 1.2. Como parte del proceso ingenieril, se exponen los principales artefactos generados por la metodología seleccionada como los requisitos de software identificados, las historias de usuario y la arquitectura empleada. Se exponen por último las conclusiones parciales sobre el trabajo realizado.

2.1 Desarrollo del modelo basado en grafo

Para la confección de la base de datos orientada a grafos se identifican los principales conceptos y relaciones que se establecen entre ellos, partiendo del análisis realizado (ver sección 1.2). En la siguiente figura se aprecia el resultado final del modelo orientado a grafo necesario para el cálculo de los indicadores antes mencionados.

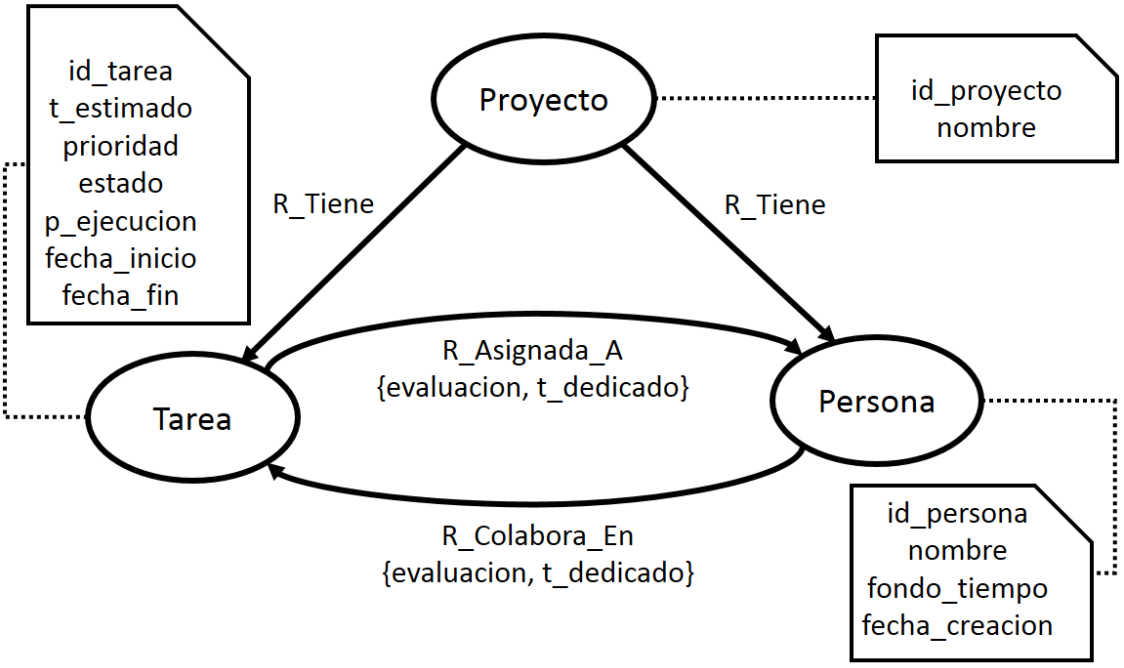

Figura 2 Modelo orientado a grafo confeccionado

Fuente: elaboración propia

Las relaciones se establecen entre los conceptos tarea, persona y proyecto, donde una tarea está asignada a una única persona, y a su vez, una persona, puede colaborar en una tarea que no le haya sido asignada, así como un proyecto tiene tareas y personas. Para realizar el cálculo de los indicadores en el nuevo modelo, se definen atributos para cada uno de los conceptos representados, como se puede apreciar en la Tabla 1.

CONCEPTO	ATRIBUTO	DESCRIPCIÓN
Proyecto	id_proyecto	Identificador numérico y único del proyecto
	nombre	Nombre identificativo del proyecto
Persona	id_persona	Identificador numérico y único de la persona
	nombre	Nombre de la persona
	fondo_tiempo	Valor numérico del fondo de tiempo del recurso humano
	fecha_creacion	Fecha de incorporación de la persona en el sistema
Tarea	id_tarea	Identificador numérico y único de la tarea
	t_estimado	Tiempo estimado de la tarea en horas
	prioridad	Prioridad de la tarea (Alta, Normal y Baja)
	estado	Estado actual de la tarea (Cerrada o no)
	p_ejecucion	Porciento de ejecución de la tarea
	fecha_ inicio	Fecha de inicio de la tarea
	fecha_fin	Fecha de cierre de la tarea

Tabla 1 Conceptos y atributos del modelo orientado a grafo propuesto

Por otro lado, las relaciones del modelo planteado se definen de la siguiente manera:

RELACIÓN	ATRIBUTO	DESCRIPCIÓN
R_Asignada_A	t_dedicado	Tiempo dedicado de la persona en la tarea
	evaluacion	Evaluación recibida de la tarea.
R_Colabora_En	t_dedicado	Tiempo total dedicado de la persona en una tarea.
	evaluacion	Evaluación recibida de la tarea.
R_Tiene		Establece la relación entre los conceptos de Proyecto – Persona y Proyecto – Tarea.

Tabla 2 Relaciones y sus atributos del modelo orientado a grafo propuesto

Luego de diseñado el modelo, se expone en secciones separadas la especificación de los elementos necesarios para realizar el cálculo de cada uno de los indicadores. Para el indicador IRHT, se identificó el siguiente conjunto de atributos:

Indicador	Atributos Requeridos
IRHT	t_estimado
	t_dedicado
	fecha_inicio
	fecha_fin
	id_persona

Tabla 3 Atributos requeridos por el indicador IRHT

Dados estos atributos, la sección del modelo requerida para el indicador es la siguiente,

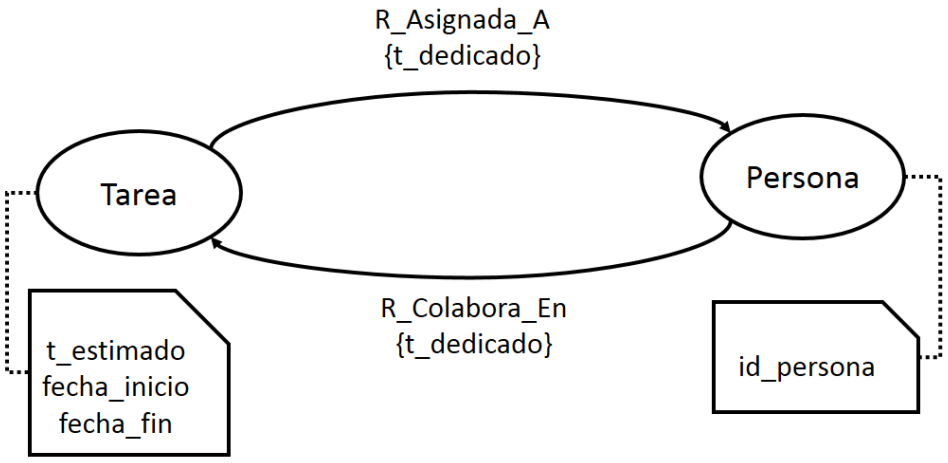

Figura 3 Sección del modelo requerido para estimar el indicador IRHT

Fuente: elaboración propia

Dada esta especificación se diseña el Algoritmo 1 "*CalcularIRHT*" para calcular el indicador IRHT. En él se hacen referencias a otros algoritmos como son:

- *CalcularPromedioTE* y *CalcularPromedioTD*: calculan el promedio de los tiempos estimados y dedicados respectivamente de las tareas asignadas y en colaboración del recurso humano.

- *CalcularDesviaciónEstándarTE* y *CalcularDesviaciónEstándarTD*: calculan la desviación estándar de los tiempos estimados y dedicados de las tareas asignadas y en colaboración del recurso humano.

Algoritmo 1 *CalcularIRHT*	
Entradas	Identificador de la persona (*id_persona*) y la fecha de corte (*fecha_corte*)
Salidas	IRHT correspondiente a la persona (*IRHT*)
Crear variables estadísticas TE y TD AVG_TE ← *CalcularPromedioTE (id_persona,fecha_corte)* AVG_TD ← *CalcularPromedioTD (id_persona,fecha_corte)* STDV_TE ←*CalcularDesviaciónEstándarTE* (*id_persona,fecha_corte*) STDV_TD ← *CalcularDesviaciónEstándarTD* (*id_persona,fecha_corte*) **Para toda** tarea T asignada o en colaboración del RH **hacer:** TE ← (*t_estimado* de T – AVG_TE) / STDV_TE TD ← (*t_dedicado* de T – AVG_TD) / STDV_TD **Fin para** *IRHT* ← CORRELACION (TE , TD) **Devolver** *IRHT*	

Algoritmo 1 CalcularIRHT (Cálculo del indicador IRHT en el modelo propuesto)

En el algoritmo "*CalcularIRHT*" puede apreciarse el uso de dos matrices de igual tamaño (TE y TD) que serán la entrada para el cálculo de la correlación de Pearson. Por otra parte, el indicador IRHE definido en (Lugo García, 2012), se determina mediante los atributos expuestos en la Tabla 4.

Indicador	Atributos Requeridos
IRHE	evaluacion
	prioridad
	estado
	fecha_inicio
	fecha_fin
	id_persona

Tabla 4 Atributos requeridos para el cálculo del indicador IRHE

A diferencia del anterior indicador, este realiza el análisis atendiendo al concepto de proyecto asociado a la persona y sus tareas. Por ello, en la sección del modelo requerido para dicho indicador, se incorporó el concepto de proyecto (ver Figura 4).

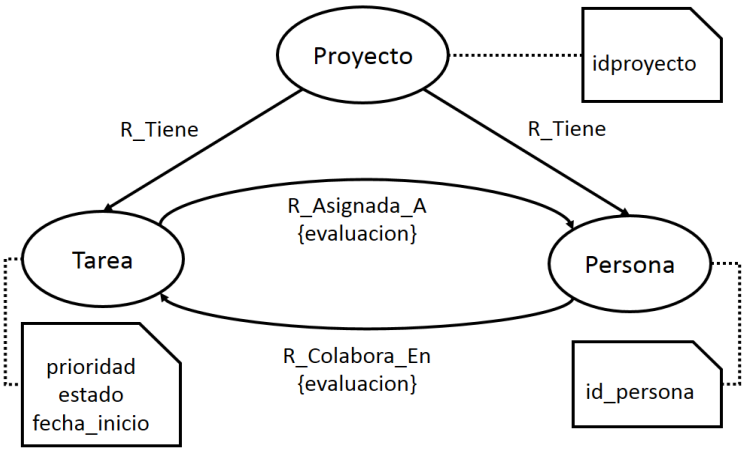

Figura 4 Sección del modelo requerido para estimar el indicador IRHE

Fuente: elaboración propia

Atendiendo a la sección del modelo para el indicador IRHE, se define el Algoritmo 2 para el cálculo de dicho indicador.

Algoritmo 2 *CalcularIRHE*	
Entradas	Identificador de la persona (*id_persona*) y la fecha de corte (*fecha_corte*)
Salidas	IRHE correspondiente a la persona (*IRHE*)

```
Inicializar CTBERH en 0
Inicializar CTC en 0
Para toda Tarea T asignada o en colaboración de la persona hacer:
    Si estado de T es Cerrada y fecha_inicio de T ≥ fecha de corte entonces
        Si evaluacion de  T es "Bien" o evaluacion  de T es "Excelente" entonces
            Si prioridad de T es "Alta" entonces CTBERH ← CTBERH + 3
            Sino si prioridad de T es "Normal" entonces CTBERH ← CTBERH + 2
            Sino CTBERH ← CTBERH + 1 Fin si
        Fin si
        Si prioridad de T es "Alta" entonces CTC ← CTC+3 Fin si
        Si prioridad de T es "Normal" entonces CTC ← CTC+2 Fin si
        Si prioridad de T es "Baja" entonces CTC ← CTC+1 Fin si
    Fin si
Fin Para
Si CTBERH = 0 o CTC = 0 entonces Devolver 0
IRHE ← CTBERH/CTC
Devolver IRHE
```

Algoritmo 2 CalcularIRHE (Cálculo del indicador IRHE en el modelo propuesto)

En cuanto al indicador IRHA, relacionado con el aprovechamiento de los recursos humanos ante las tareas asignadas, se muestra en la Tabla 5:

Indicador	Atributos requeridos
IRHA	Tiempo estimado de la tarea
	Fondo de tiempo de la persona
	Fecha de inicio de la tarea
	Fecha fin de la tarea
	Identificador de la persona

Tabla 5 Atributos requeridos para el cálculo del indicador IRHA

Una vez definidos los atributos requeridos para determinar el indicador IRHA, la sección del modelo indispensable para calcular dicho indicador se muestra en la Figura 5**¡Error! Marcador no definido.** de la siguiente forma:

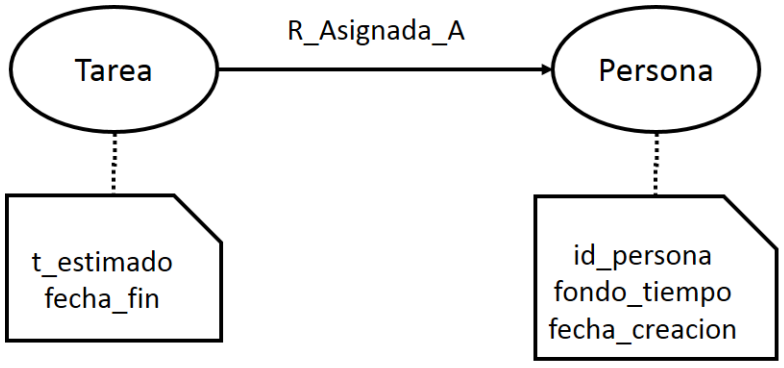

Figura 5 Sección del modelo requerido para el cálculo del indicador IRHA

Fuente: elaboración propia

Para la determinación del indicador en el modelo propuesto, se define el siguiente algoritmo, representado en pseudocódigo para mayor comprensión del mismo.

Algoritmo 3 *CalcularIRHA*	
Entradas	Identificador de la persona(*id_persona*) y fecha de corte (*fecha_corte*)
Salidas	IRHA de la persona (*IRHA*)
Inicializar *TTP* en 0 TTD ← *TiempoDisponibleDePersona(id_persona)* **Para toda** tarea *T* asignada a la persona **hacer:** **Si** estado de *T* es Abierta y fecha de fin de T < fecha_corte **entonces**	

```
            TTP ← TTP + EjecucionPlanificada (T, fecha_corte)
        Fin si
Fin para.
IRHA ← TTP/TTD.
Devolver IRHA.
```

Algoritmo 3 CalcularIRHA (Cálculo del indicador IRHA en el modelo propuesto)

En el anterior algoritmo, el tiempo disponible de una persona es determinado a través del fondo de tiempo total utilizado por la persona desde su registro en el sistema. Por último, el indicador IRHF, que refiere el avance de la ejecución real de las tareas en cuanto a la planificación, los atributos requeridos se expresan en la Tabla 6.

Indicador	Atributos requeridos
IRHF	t_estimado
	p_ejecucion
	fecha_inicio
	fecha_fin
	id_persona

Tabla 6 Atributos requeridos para el cálculo del indicador IRHF

La sección del modelo requerido para estimar el indicador IRHF contiene los atributos que se reflejan en la Tabla 6, quedando expresado en la Figura 6.

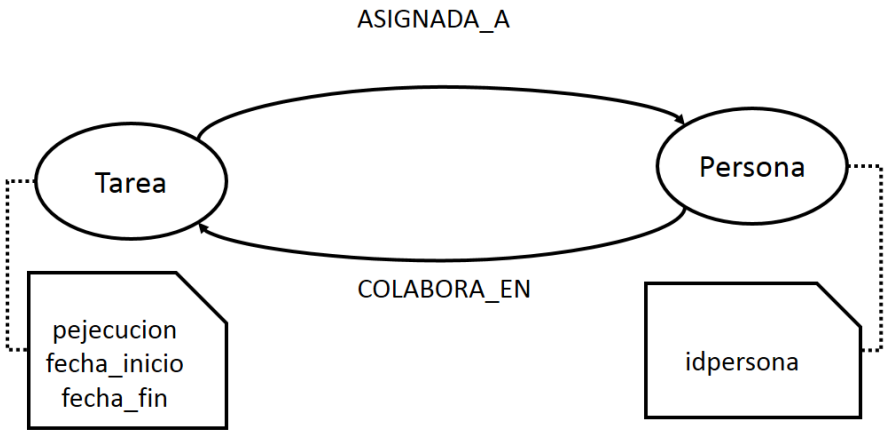

Figura 6 Sección del modelo requerido para el cálculo del indicador IRHF

Fuente: elaboración propia

Con objetivo de calcular el indicador IRHT a partir del modelo propuesto, y teniendo en cuenta la sección del mismo mostrado en la Figura 6, se diseña el Algoritmo 4.

Algoritmo 4 *CalcularIRHF*	
Entradas	Identificador de la persona (*id_persona*) y fecha de corte (*fecha_corte*)
Salidas	IRHF correspondiente a la persona (*IRHF*)
SERT = 0; *SEPT* =0. P ←*BuscarPersonaConID*(*id_persona*) **Para toda** tarea *T* asignada o en colaboración de P **hacer:** **Si** *fecha_creacion* de *T* es menor o igual que *fecha_corte* **entonces** *SERT* ← *SERT* + (t_*estimado* de T * p_*ejecucion* de T) *SEPT* ← *SEPT* + *EjecucionPlanificada (*T*, fecha_corte)* **Fin si** **Fin para** *IRHF* ←*SERT/SEPT.* **Devolver** *IRHF*.	

Algoritmo 4 CalcularIRHF (Cálculo del indicador IRHF en el modelo propuesto)

En el Algoritmo 4, son utilizadas las funcionalidades "*BuscarPersonaConID*", que realiza la búsqueda en el grafo del nodo correspondiente a la persona por su identificador; y "*EjecucionPlanificada*" (ver Algoritmo 5), que determina el porcentaje de cumplimiento de una tarea dada una fecha de corte, y sus fechas de inicio y fin. Las operaciones de resta entre fechas que se utilizan, determinan la diferencia en días entre una fecha y otra.

Algoritmo 5 *EjecucionPlanificada*	
Entradas	Tarea (*T*) y la fecha de corte (*fecha_corte*)
Salidas	Ejecución planificada de la tarea (*EP*)
Inicializar EP = 0; **Si** *fecha_fin* de T ≤ *fecha_corte* **entonces** EP ← 100 **Sino si** *fecha_corte* de T ≤ *fecha_inicio* de T **entonces** EP ← (*p_ejecucion* de T / (100+*p_ejecucion* de T) * 100 **Sino si** *fecha_inicio* de T ≤ *fecha_corte* y *fecha_fin* de T > *fecha_corte* **entonces** EP ← ((*fecha_corte* – *fecha_inicio* de T) / (*fecha_fin* de T – *fecha_inicio* de T) * 100) **Fin si** **Retornar** EP	

Algoritmo 5 EjecucionPlanificada

2.2 Desarrollo de vistas sobre el modelo

Partiendo del modelo orientado a grafos propuesto, son confeccionadas tres vistas para el apoyo a la toma de decisiones:

1. Personas de un proyecto
2. Proyectos de una persona.
3. Tareas abiertas de una persona.

La vista “Personas de un proyecto”, muestra las personas que pertenecen a un proyecto especificado, así como las tareas que tiene pendientes en dicho proyecto. La representación de esta vista, permite comprobar el balance y distribución de las tareas en un proyecto gestionado, así como las personas que tienen una mayor carga de trabajo con respecto a otras. En la Figura 7 puede apreciarse su representación, concebida a través de la interfaz web de administración de la herramienta Neo4j.

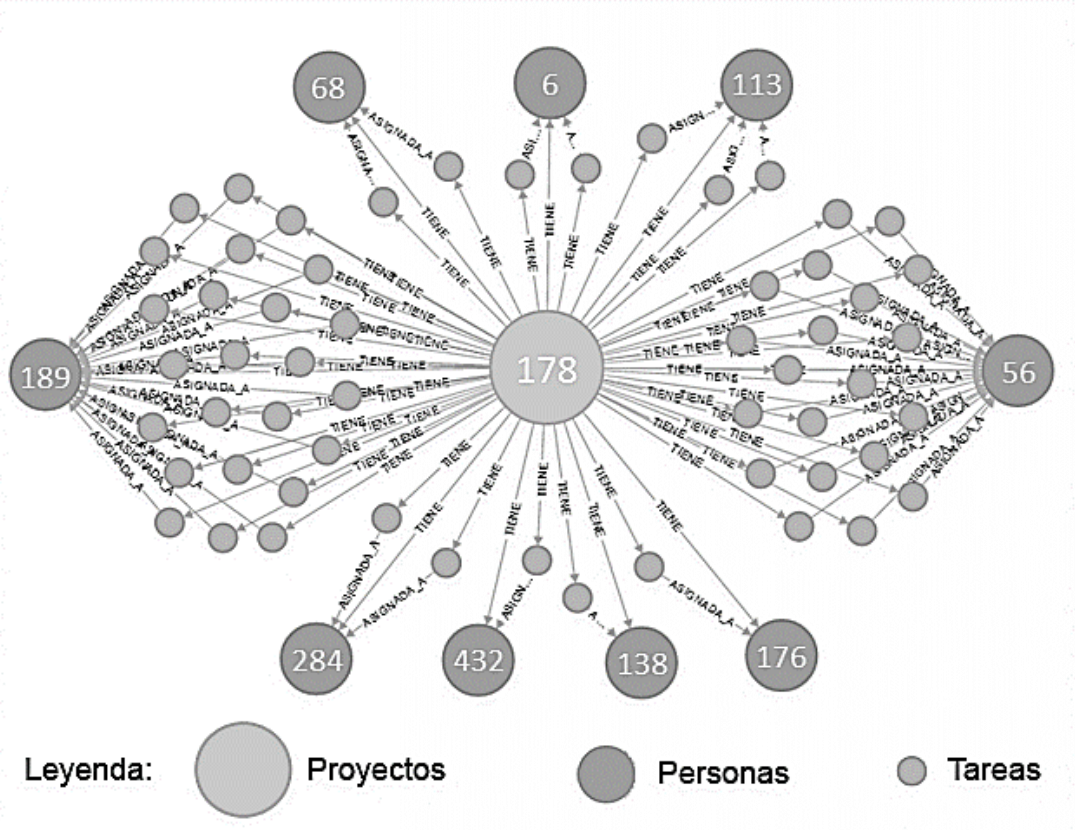

Figura 7 Vista "Personas de un proyecto"

Fuente: elaboración propia

Como puede apreciarse en el ejemplo de la Figura 7, las personas (nodos de color verde) con identificador 189 y 56, tienen un mayor cúmulo de tareas (nodos de color rojo) en el proyecto analizado (nodo de color morado).

La vista “Proyectos de una persona” muestra las relaciones que se establecen entre los diferentes proyectos a los que pertenece una persona, así como las tareas que posee en cada uno de los proyectos mencionados. Permite ver la asignación de tareas que tiene una persona en los proyectos a los que pertenece, y con ello identificar sobrecarga o desaprovechamiento del recurso humano.

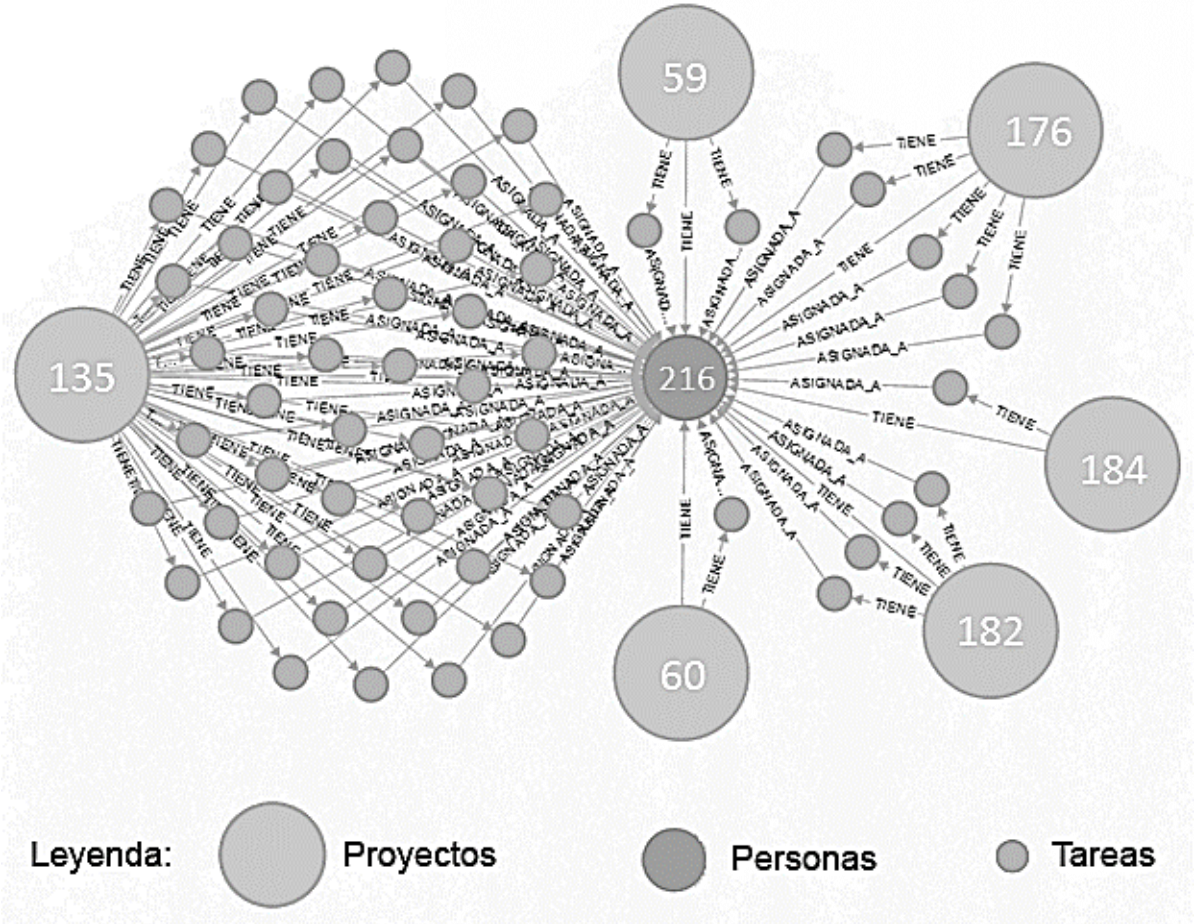

Figura 8 Vista "Proyectos de una persona"

Fuente: elaboración propia

Un ejemplo de esta vista puede apreciarse en la Figura 8, donde la persona con identificador 216, presenta un desbalance en las tareas asignadas en cada proyecto. Se evidencia mayor actividad en el proyecto con identificador 135.

La vista “Tareas abiertas de una persona”, es elaborada como una extensión de la vista anterior. Para ello, se muestran solamente las tareas abiertas que tiene una persona en los respectivos proyectos a los que pertenece. Ello permite identificar con rapidez qué tareas no ha concluido en el momento del análisis. En la Figura 9 puede apreciarse un ejemplo de la vista propuesta.

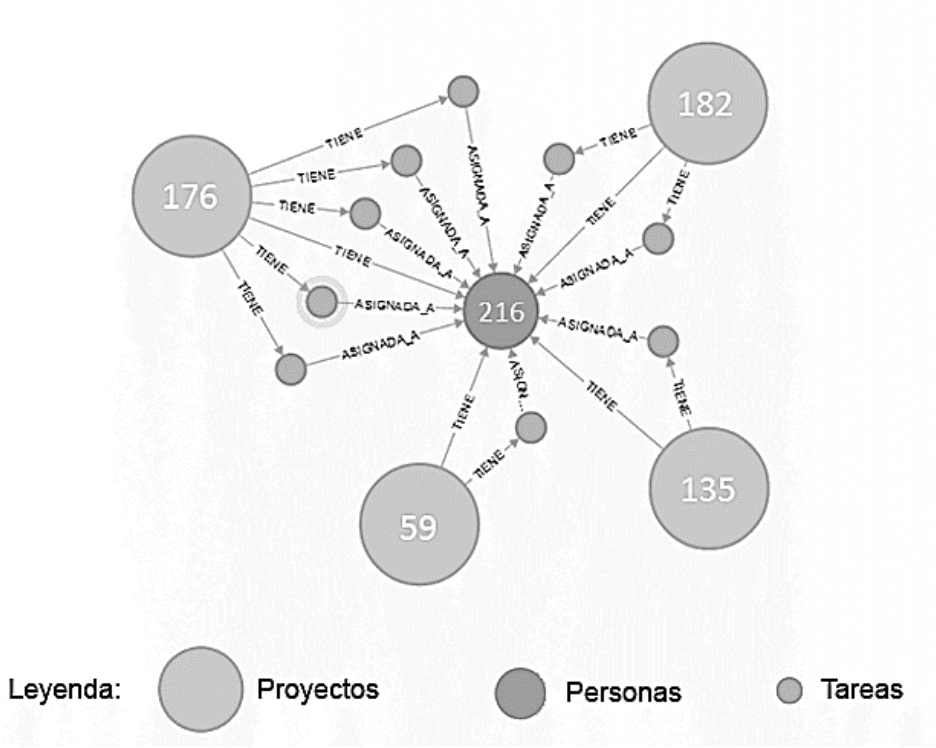

Figura 9 Vista "Tareas abiertas de una persona"

Fuente: elaboración propia

2.3 Realización de tareas ingenieriles

En el siguiente acápite, se expone una descripción de la herramienta propuesta, así como los principales conceptos que se tuvieron en cuenta en su desarrollo Se realiza un levantamiento de requisitos junto al cliente dando como resultado las historias de usuario que define la metodología de desarrollo

seleccionada, así como los requisitos no funcionales que debe garantizar el sistema en cuanto a rendimiento, hardware, software, soporte, interfaz y usabilidad.

Con motivo de realizar una eficiente distribución de las responsabilidades ante el trabajo a realizar, la metodología XP propone una serie de actores que intervienen en el proceso de creación del software. Los roles que propone la metodología, así como la persona asignada en el presente trabajo se muestran en la Tabla 7.

Roles que define la metodología XP	Encargado
Cliente	José Alejandro Lugo García
Programador	Rigoberto Peña Cabrera
Consultor	Maiko Hernández Martínez
Tester (Encargado de pruebas)	Maiko Hernández Martínez
Tracker (Encargado de seguimiento)	Maiko Hernández Martínez
Entrenador (Coach)	Rigoberto Peña Cabrera
Gestor (Big Boss)	Rigoberto Peña Cabrera

Tabla 7 Asignación de los roles que propone XP

2.3.1 Descripción del sistema

Teniendo en cuenta la problemática planteada, se definió desarrollar una aplicación informática con los siguientes procesos:

- Establecer conexión con la fuente de datos del sistema Xedro-GESPRO.
- Realizar la transformación de los datos obtenidos del modelo relacional al modelo orientado a grafo, o utilizar una transformación previamente realizada.
- Realizar análisis sobre el grafo confeccionado a través de vistas predefinidas.
- Calcular los indicadores analizados en la sección 1.2 del capítulo 1.

Por lo antes expuesto, el sistema cuenta con tres escenarios fundamentales: transformación, cálculo de indicadores y análisis de vistas. Dentro de la transformación se tiene en cuenta los aspectos de configuración de la conexión con la fuente de datos origen; autenticación del usuario; selección de la base de datos, creación de la base de datos orientada a grafos teniendo en cuenta su ubicación y sus propiedades. Una vez establecidas todas las configuraciones, se procede al proceso de transformación,

que mediante un proceso ETL extrae la información, proporciona los elementos del grafo (nodos, relaciones y sus atributos), persistiéndolos en la fuente de datos destino. Durante esta actividad, se realizan notificaciones sobre los errores de transformación detectados, mostrando un informe detallado con los eventos ocurridos luego de completada la operación.

Para el escenario del cálculo de indicadores, el usuario tiene la posibilidad de seleccionar de forma manual los indicadores a calcular, así como el recurso humano sobre el cual se realiza dicho cálculo. Es de interés en esta sección el tiempo computacional que requieren los indicadores en el modelo orientado a grafo, para luego realizar comparaciones con el modelo relacional sobre el que se calculan actualmente.

En cuanto al tercer escenario se representan diferentes vistas del grafo confeccionado, que contribuyan a la toma de decisiones en el ámbito de los recursos humanos por parte de los decisores. Se permite filtrar algunas propiedades o atributos de los conceptos del grafo, que faciliten la búsqueda de información más precisa.

2.3.2 Requisitos funcionales

El proceso de captura de requisitos es una etapa de suma importancia dentro del proceso de desarrollo de software. Este se encarga de analizar las necesidades del cliente, describirlas en detalle para conformar el sistema de la manera más precisa, cumpliendo siempre con las especificaciones deseadas. En el presente trabajo, se identificaron los requisitos funcionales que se muestran en la Tabla 8, agrupándolos por los escenarios principales del sistema descrito.

Número	Escenario	Nombre
1	Transformación	Establecer conexión con sistema Xedro-GESPRO
2		Crear base de datos orientada a grafos
3		Detectar transformaciones realizadas
4		Transformar datos
5		Controlar proceso de transformación
6		Notificar eventos de transformación

7		Mostrar informe de transformación
8		Guardar informe de transformación
9	Cálculo de indicadores	Calcular indicador IRHT a persona
10		Calcular indicador IRHE a persona
11		Calcular indicador IRHA a persona
12		Calcular indicador IRHF a persona
13		Calcular indicador IRHT general
14		Calcular indicador IRHE general
15		Calcular indicador IRHA general
16		Calcular indicador IRHF general
17		Calcular indicadores
18		Mostrar resultados de cálculo de indicador
19	Vistas	Mostrar vista Personas de Proyecto
20		Mostrar vista Proyectos de Persona
21		Mostrar vista Tareas abiertas de Persona
22		Filtrar campos de vista
23		Guardar imagen de vista

Tabla 8 Requisitos funcionales identificados

2.3.3 Requisitos no funcionales

Para lograr mayor aceptación por parte del usuario final del sistema a conformar, se tienen en cuenta también los requisitos no funcionales. Según (Stellman & Greene, 2005) son definidos como requerimientos que deben dar respuesta a la operatividad del sistema final. Se encuentran divididos en dos categorías fundamentales:

- Cualidades de ejecución, como la seguridad y facilidad de uso, que son observables en tiempo de ejecución.
- Cualidades de evolución, como la capacidad de prueba, mantenimiento, ampliación y escalabilidad, que se enfoca en la estructura estática del sistema.

Para el siguiente trabajo, atendiendo a lo anterior, se especifican los requisitos no funcionales de acuerdo a las categorías que propone la norma ISO 9126-1 (ISO/IEC, 2000) en la Tabla 9.

Categoría	Número	Descripción
Seguridad	RNF1	El sistema debe garantizar el acceso a la fuente de datos de Xedro – GESPRO mediante autenticación (usuario y contraseña).
	RNF2	El sistema debe utilizar las credenciales de autenticación solo para el acceso a la fuente de datos del sistema Xedro – GESPRO, impidiendo su almacenamiento.
Hardware	RNF3	El sistema debe proporcionar las características de hardware a la estación de trabajo que permita su correcto funcionamiento, como son: • Memoria RAM de 4 GB a 6GB. • Microprocesador con velocidad de procesamiento de 3.0 GHz. • Almacenamiento de 300 MB, con una tasa de crecimiento de un 20% por año.
Software	RNF4	El sistema debe contar con la máquina virtual de java (JVM 8 actualización 20) instalada.
Interfaz de usuario	RNF5	El sistema debe presentar las mismas dimensiones (800x500 pixeles) para todas sus interfaces de usuario.
	RNF6	El sistema debe poseer los botones de navegación (Siguiente, Atrás, Salir) en la parte inferior de la interfaz de usuario.
	RNF7	El sistema debe proporcionar campos de selección en las interfaces de usuario siempre que sea posible.

Soporte	RNF8	El sistema debe generar registros en cada actividad que pueda tener una incidencia negativa para el usuario, como son: • Errores en la obtención de los datos a transformar desde el sistema Xedro – GESPRO. • Errores en la autenticación. • Alertas sobre existencia de relaciones o nodos en la conformación del modelo orientado a grafos. • Errores en la importación de las imágenes de las diferentes interfaces de usuario.
	RNF9	El sistema debe garantizar la disponibilidad de la ayuda en cada interfaz de usuario mediante un botón en la región inferior izquierda.
Rendimiento	RNF10	El sistema debe realizar la transformación de los datos en un intervalo de 10 a 20 registros por segundo.
	RNF11	El sistema debe realizar el cálculo de los indicadores en un tiempo menor que el empleado en el sistema Xedro – GESPRO.

Tabla 9 Requisitos no funcionales definidos

2.3.4 Fase de Exploración

La metodología de desarrollo XP comienza con la fase de exploración, donde los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizaron en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo la versión 1.0. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología (Beck & Andres, Extreme Programming Explained: Embrace Change, 1999).

Historias de usuarios

Las historias de usuario (HU) representan un requisito (o agrupación de ellos) de software escrito en una o dos frases utilizando el lenguaje común del usuario. Proporcionan los detalles sobre la estimación del riesgo y el tiempo necesario para la implementación de dicha HU. Además son una forma simple de administrar los requisitos de los usuarios, lo que permite responder rápidamente a los cambios.

Durante el análisis en la fase de exploración fueron identificadas nueve HU, que recogen los requisitos previamente identificados (ver Tabla 8), además proporcionan una idea al equipo de desarrollo sobre cómo debe ser su posterior implementación.

Para realizar una correcta planificación de la implementación se debe tener en cuenta la prioridad de cada HU. Partiendo de las necesidades de desarrollo a cada una se le asigna una clasificación que puede ser:

- Alta: fundamentales para el desarrollo del sistema.
- Media: poseen funcionalidades necesarias pero no imprescindibles para el sistema.
- Baja: constituyen funcionalidades que sirven de ayuda al control de los elementos asociados al equipo de desarrollo y a la estructura.

Para conocer la dificultad y posibles errores durante la implementación de cada HU, el equipo de desarrollo clasifica el riesgo en:

- Alto: cuando la implementación de la HU se considera la posible existencia de errores que lleven a la inoperatividad del código.
- Medio: cuando puedan aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.
- Bajo: cuando puedan aparecer errores que pueden ser corregidos con relativa facilidad sin que ocasionen prejuicios para el desarrollo del software.

El tiempo determinado para cada HU conocido como puntos estimados, se establece en semanas ideales (40 horas) a lo que se puede concluir que: 0.1 equivale a medio día de trabajo (4 horas). Las HU son representadas mediante tablas con las siguientes secciones:

- Número: número de la HU el cual es incremental en el tiempo.
- Nombre de la HU: identificador de la HU que se describe entre los desarrolladores y el cliente.
- Usuario: rol del usuario que realiza la funcionalidad.
- Prioridad en negocio: prioridad de acuerdo a la necesidad de desarrollo.
- Riesgo en desarrollo: riesgo de acuerdo a la posibilidad de ocurrencia de errores en el proceso de implementación de la HU.
- Iteración asignada: número de la iteración donde se va a desarrollar la HU.
- Puntos estimados: tiempo de desarrollo de la HU.
- Descripción: breve descripción de la HU.

- Observaciones: alertas o señalamientos del sistema.

Como resultado de un intercambio con el cliente, fueron confeccionadas un total de nueve HU:

1. Verificar conexión.
2. Transformar datos.
3. Mostrar informe de transformación.
4. Calcular indicadores a nivel de persona.
5. Calcular indicadores a nivel general.
6. Calcular indicadores a nivel global.
7. Mostrar vista Proyectos de una Persona.
8. Mostrar vista Personas de un Proyecto.
9. Mostrar vista Tareas abiertas de Persona.

A continuación se expone un ejemplo de las HU conformadas para el sistema desarrollado, teniendo en cuenta los diferentes atributos analizados para su confección, así como los requisitos funcionales involucrados en ella. El resto de las HU elaboradas se encuentran presentes en el Anexo 2.

Historia de usuario	
Número: 1	**Nombre de la HU:** Verificar conexión
Requisitos relacionados: RF1, RF3	**Usuario:** Administrador
Prioridad en negocio: alta	**Riesgo en desarrollo:** bajo
Puntos estimados: 0.2	**Iteración asignada:** 1
Descripción: en la interfaz se muestra un formulario con los parámetros para establecer la conexión a la base de datos del sistema Xedro-GESPRO: dirección del servidor, puerto, usuario, contraseña y nombre de la base de datos. Una vez introducidos se permite probar la conexión, informando si se estableció correctamente o si ocurrió algún error. En otra sección se requiere el nombre de la base de datos orientada a grafos a conformar. Una vez introducido todos los campos de la interfaz, se activa el botón de siguiente para avanzar a la siguiente interfaz.	
Observaciones: en caso de que alguno de los parámetros requeridos no sea válido se muestra un mensaje que lo indica.	
Prototipo de interfaz de usuario	

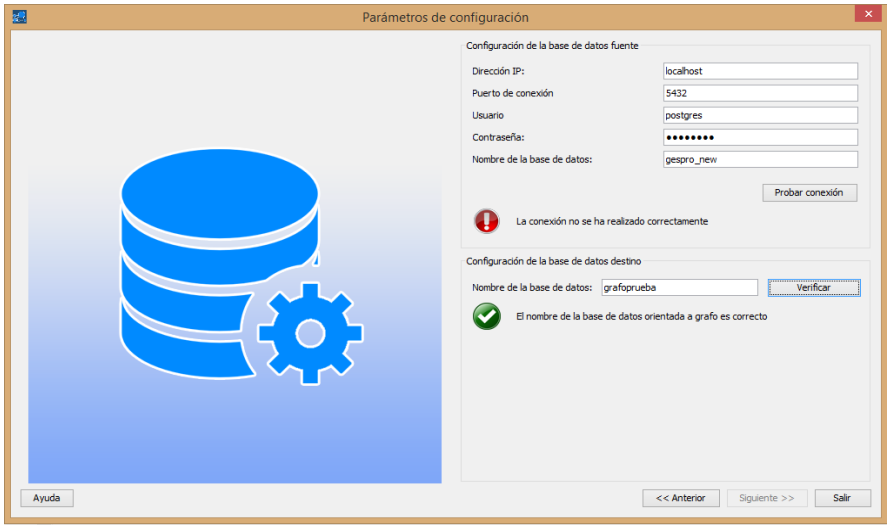

Tabla 10 Historia de usuario Verificar conexión

2.3.5 Fase de Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario. Luego, los programadores del equipo de trabajo estiman el esfuerzo necesario para cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente para la realización del resto de las versiones.

Plan de entrega

Teniendo en cuenta el esfuerzo asociado a las historias de usuario y las prioridades del cliente se define una entrega (*release*) que sea de valor y que tenga una duración de pocos meses. La entrega es dividida en iteraciones de no más de 3 semanas, asignando a cada iteración un conjunto de historias de usuario que serán implementadas (Sanchez , Letelier Torres, & Canós Cerdá, 2003). En la Tabla 11 se aprecia la planificación realizada para el presente trabajo.

Iteración	Fecha de inicio	Fecha de entrega
1	25/02/2015	17/03/2015
2	18/03/2015	08/04/2015
3	09/04/2015	16/04/2015

Tabla 11 Plan de entregas

2.3.6 Fase de Iteraciones

Esta fase dispone las iteraciones para conformar el sistema propuesto. El plan de entrega se compone de iteraciones de no más de cuatro semanas. En la primera iteración se elabora la arquitectura inicial del sistema a utilizar durante el desarrollo del proyecto.

Las HU son seleccionadas de acuerdo al orden preestablecido para cada entrega, son implementadas a partir del orden establecido por el cliente comenzando por las de mayor prioridad para maximizar el valor del negocio. Al finalizar la última iteración el sistema se encuentra listo para entrar en producción y para cada HU se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, verificando así, que subsiguientes iteraciones no han afectado a las anteriores.

El objetivo principal de la iteración 1 es la implementación de las historias de usuarios seleccionadas de mayor prioridad, de esta forma se obtienen las primeras funcionalidades vitales para el desarrollo del sistema donde se establece la conexión con la base de datos relacional, se realiza una trasformación de los datos a una base de datos orientada a grafo y se notifican los resultados obtenidos de la transformación, así como la primera versión del sistema.

Para la segunda iteración se implementan las historias de usuario de prioridad alta correspondientes al cálculo de los indicadores para determinar el rendimiento de los recursos humanos, además se corrigen los errores de las HU de la iteración anterior. De esta forma se tiene la segunda versión del producto. Por último, en la tercera iteración se implementan las HU relacionadas con las vistas y funcionamiento del sistema que poseen una prioridad baja, donde algunas de estas ya están predefinidas y presentan la peculiaridad que se pueden realizar diferentes configuraciones de las mismas.

En la metodología XP, para que exista mayor organización a la hora de desarrollar un software, es confeccionado el plan de duración de las iteraciones. El mismo tiene como objetivo fundamental mostrar la iteración en que será implementada cada HU según el orden correspondiente a cada una de ellas, así como el tiempo que se ha destinado para cada iteración.

Iteración	Orden de la HU a implementar	Duración (semanas)
1	1. Verificar conexión	2.7
	2. Transformar datos	

	3. Notificar resultados de transformación	
2	4. Calcular indicador IRHT	3
	5. Calcular indicador IRHE	
	6. Calcular indicador IRHA	
	7. Calcular indicador IRHF	
3	8. Mostrar vistas Proyectos de Persona	0.4
	9. Mostrar vistas Personas de un Proyecto	0.4
	10. Mostrar vista Tareas abiertas de una Persona	0.2
Total		**6.7**

Tabla 12 Plan de iteraciones

2.3.7 Fase de Producción

En la fase de producción se requieren de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea desplegado al entorno del cliente. Al mismo tiempo se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios en esta fase. Como actividades fundamentales se encuentran el diseño de la arquitectura del sistema, así como la selección de los patrones de diseño, la elaboración de las tarjetas de Clase – Responsabilidad – Colaboración (CRC) y las pruebas constantes a cada HU.

Arquitectura del sistema

Para el presente trabajo, dadas las necesidades de realizar un sistema que interactúe con el usuario de forma dinámica y constante, así como el trabajo con bases de datos de distintas tecnologías y el manejo de un proceso ETL para la transformación de la información, se decide optar por el estilo arquitectónico llamada y retorno. Como familia de este se encuentran las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas (Bass, Clements, & Kazman, 1998).

Fue seleccionada la arquitectura basada en capas, pues soporta un diseño distribuido en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. Admite muy naturalmente optimizaciones y refinamientos, así como la separación lógica de las funcionalidades de cada componente interno de una capa (Garlan & Shaw, 1994).

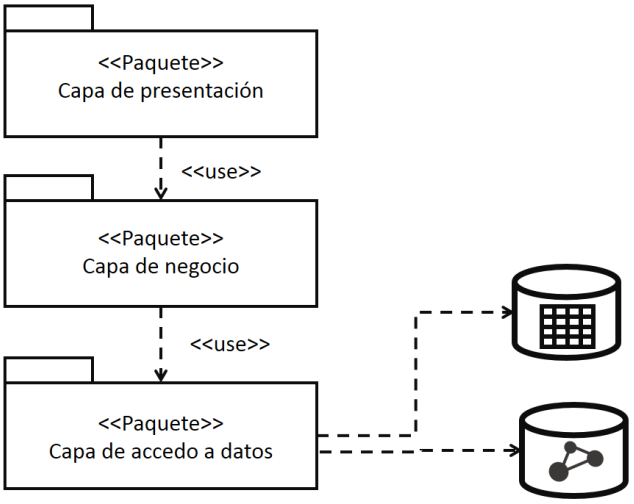

Figura 10 Arquitectura definida para la herramienta a desarrollar

Fuente: elaboración propia

Las capas propuestas para la arquitectura del sistema a desarrollar y su propósito se expresan a continuación:

La capa de presentación, que se ocupa de la interacción con el usuario, posee las diferentes interfaces de la solución. En la siguiente figura se muestran las relaciones entre los componentes definidos:

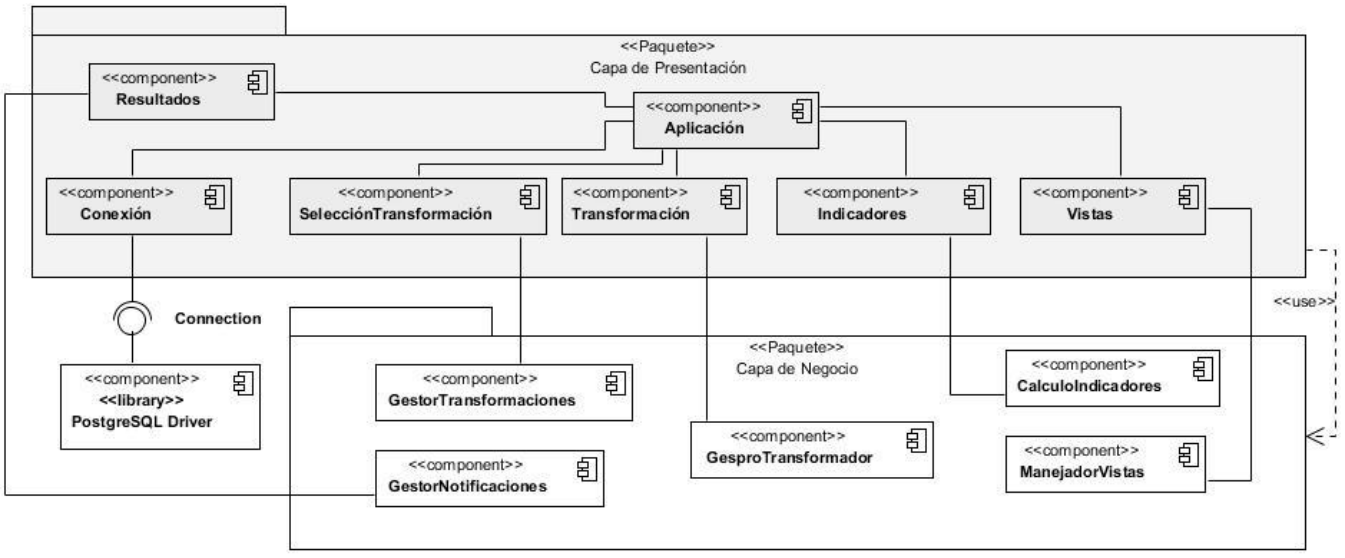

Figura 11 Componentes definidos en la capa de presentación

Fuente: elaboración propia

Los componentes definidos en la capa de presentación son los siguientes:

- Componente *Resultados*: maneja la interfaz de los resultados obtenidos luego de la transformación de la información del sistema Xedro-GESPRO hacia la base de datos orientada a grafos. Utiliza al componente *GestorNotificaciones* de la capa de negocio para la administración de las notificaciones y el registro de los errores ocurridos, así como su almacenamiento local.
- Componente *Conexión*: maneja la interfaz de conexión con el sistema de Xedro-GESPRO, especificando cada uno de los parámetros de conexión (usuario, contraseña, nombre de la base de datos, dirección del servidor y puerto). Verifica la conexión con las configuraciones establecidas, notificando las causas de las excepciones que se arrojen. Como muestra la Figura 11, utiliza la API de PostgreSQL con las funcionalidades antes descritas.
- Componente *SelecciónTransformación*: permite al usuario utilizar una base de datos orientada a grafo ya conformada, o crear una nueva, utilizando para ello una interfaz que permita al usuario seleccionar una de estas opciones. Hace uso del componente *GestorTransformaciones* para la verificación en el sistema de archivos de las bases de datos orientadas a grafo existentes.
- Componente *Transformación*: maneja la interfaz de transformación, notificando los diferentes eventos que ocurren en tiempo real en dicho proceso, obteniendo las notificaciones a partir del componente *GesproTransformador* de la capa de negocio.
- Componente *Indicadores*: posee la interfaz donde se realiza el cálculo de los indicadores analizados en la sección 1.2 del capítulo 1, permitiendo al usuario definir los parámetros para el cálculo. Se relaciona con el componente *CalculoIndicadores* de la capa de negocio, obteniendo de este los resultados y notificaciones de las funcionalidades que realiza.
- Componente *Vistas*: maneja la interfaz de las vistas predefinidas sobre el grafo confeccionado, permite interactuar con los diferentes atributos de los conceptos del grafo visualizado. Adquiere la información a partir del componente *ManejadorVistas* de la capa de negocio.
- Componente *Aplicación*: contiene la interfaz inicial del sistema, e integra los restantes componentes de la capa de presentación.

La capa de negocio contiene el conjunto de componentes que manejan los procesos de transformación de la información contenida en el sistema Xedro – GESPRO; el cálculo de los indicadores a través del modelo propuesto, la notificación de las actividades realizadas en los anteriores escenarios, así como

en la visualización de vistas sobre el grafo que permitan la toma de decisiones. Esto implica realizar un procesamiento de la fuente de datos, la validación de cualquier dato proveniente de la capa de presentación y la ejecución de algoritmos específicos. En la Figura 12 puede apreciarse la representación en el lenguaje de modelado UML de dicha capa.

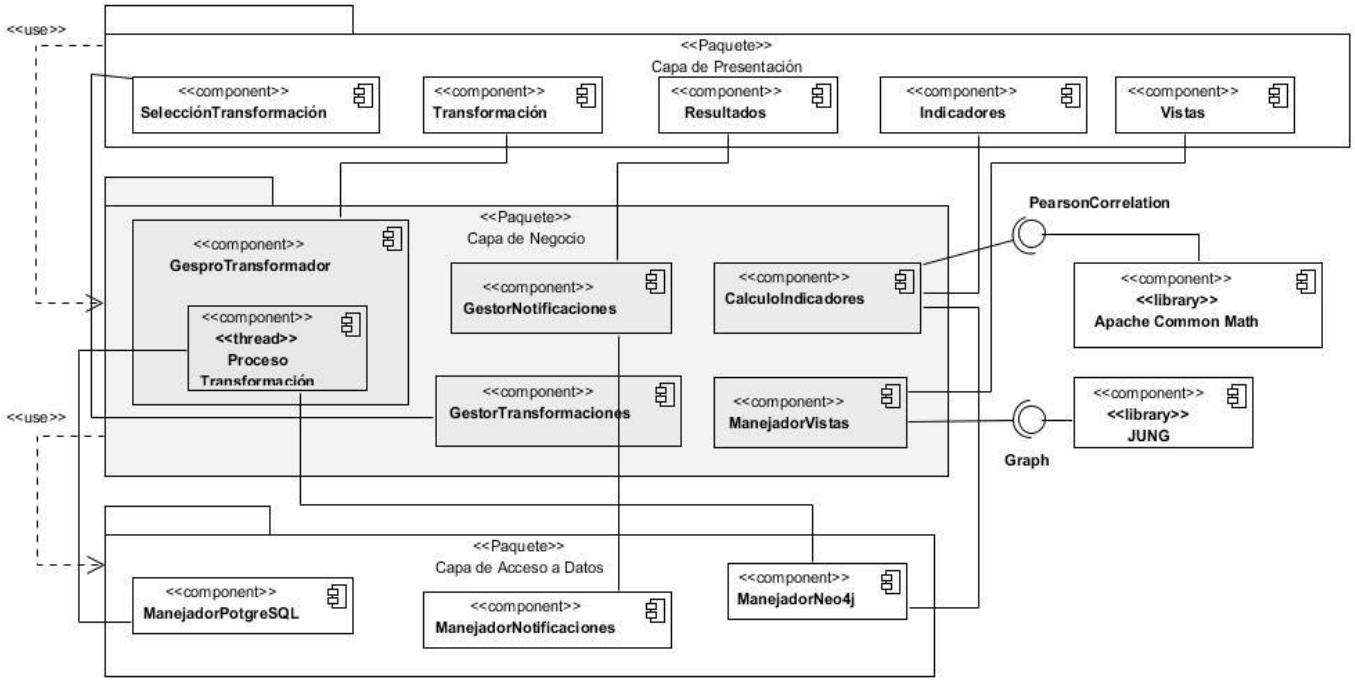

Figura 12 Componentes definidos en la capa de negocio

Fuente: elaboración propia

Dentro de esta capa, se encuentran definidos los siguientes componentes:

- Componente *GesproTransformador*: maneja todo el proceso de transformar los datos del modelo relacional existente en el modelo orientado a grafo propuesto en la sección 2.1. Para ello, maneja un hilo de tarea (componente *ProcesoTransformación*) correspondiente al proceso ETL a realizar sobre la fuente de datos origen. Además de lo anterior, maneja las conexiones establecidas con la base de datos de PostgreSQL y la orientada a grafos de Neo4j.
- Componente *ProcesoTransformación*: realiza el proceso de extracción, carga y transformación de los datos de un modelo a otro.

- Componente *GestorNotificaciones*: controla en el sistema el manejo de las notificaciones realizadas tanto en las operaciones de transformación (resúmenes), como en los eventos de error del sistema.
- Componente *GestorTransformaciones*: administra las transformaciones realizadas.
- Componente *CalculoIndicadores*: realiza el cálculo de los indicadores analizados en la sección 1.2, a través de los algoritmos diseñados sobre el modelo orientado a grafo, ambos propuestos en el presente capítulo. Utiliza para ello la biblioteca Apache Common Math para la determinación de la correlación de Pearson.
- Componente *ManejadorVistas*: gestiona la generación visual de los grafos solicitados por el usuario, utilizando para ello el conjunto de bibliotecas de JUNG.

La capa de acceso a datos, representada en la Figura 13, es la capa intermedia entre los datos físicos y la capa de negocio. Se ocupa de comunicar la herramienta con la base de datos del sistema Xedro-GESPRO y la base de datos orientada a grafo a confeccionar mediante una transformación. Además, proporciona las vías para la generación de registros requeridos por capas superiores.

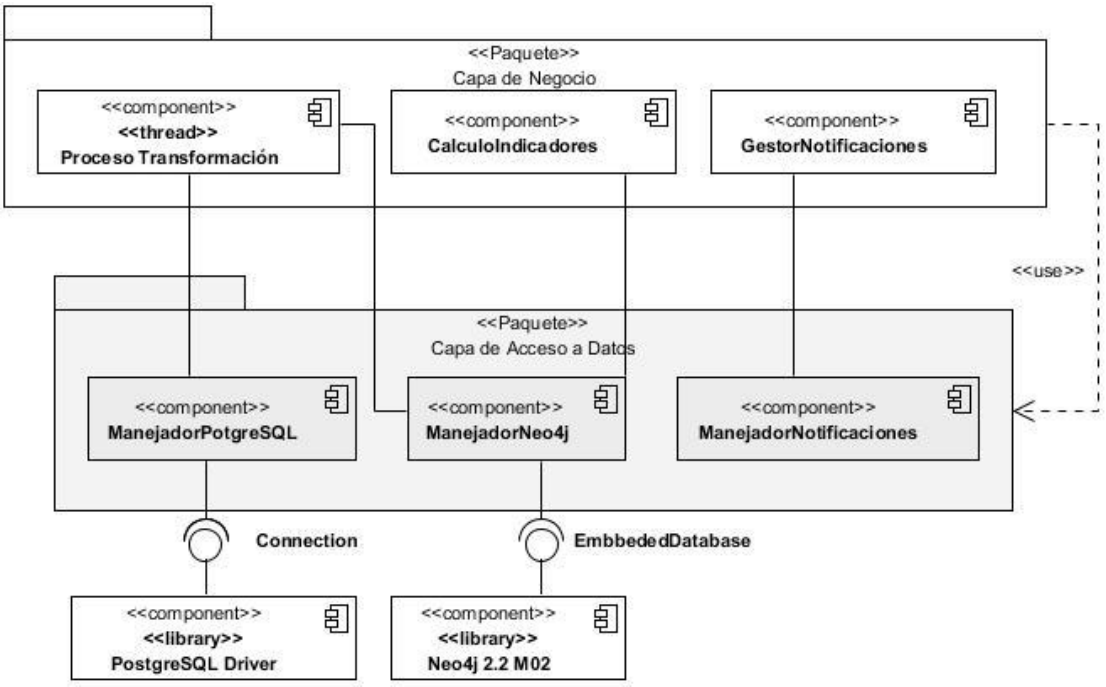

Figura 13 Componentes de la capa de acceso a datos

Fuente: elaboración propia

Los componentes definidos en esta capa son los siguientes:

- Componente *ManejadorPostgreSQL*: realiza todo el procesamiento de información referente a la base de datos del sistema Xedro-GESPRO. Contiene las funcionalidades de consulta de la información a transformar, utilizando el lenguaje de consulta de PostgreSQL.
- Componente *ManejadorNeo4j*: maneja la base de datos orientada a grafos del sistema, conteniendo todas las operaciones de consulta y modificación de los conceptos del grafo. Como se definió en la sección 1.5, la tecnología a emplear para la consulta de información en ella es mediante el propio API de Neo4j contenidas en la biblioteca Neo4j2.2 M02.
- Componente *ManejadorNotificaciones*: realiza la persistencia de las notificaciones ocurridas en una transformación, así como el registro de los eventos y errores del propio sistema desarrollado.

Patrones de diseño

En la solución propuesta fueron utilizados los patrones de diseño con el objetivo de evitar problemas previamente conocidos en el diseño de las clases del sistema y su implementación (Alexander, Ishikawa, & Silverstein, 1977). Entre los GRASP (Larman, UML y patrones, 1999) se reutilizó el patrón experto, mientras que de los GoF (Gamma, Helm, Johnson, & Vlissides, 1995) el patrón instancia única, observador e iterador.

El patrón experto (*Expert* en inglés) se utilizó para responsabilizar cada clase de acuerdo a la información que domina. Un ejemplo del uso de este patrón se expone en las clases de entidad (ver Figura 15) para la representación de los elementos del grafo. Cada clase de entidad, como infiere el patrón, es responsable del dominio de sus funcionalidades y propiedades.

Para la creación de los objetos de notificación, su almacenamiento y uso posterior, fue utilizado el patrón creador en la clase ManejadorNotificaciones (ver Figura 14). Como define el patrón, dicha clase contiene los elementos de tipo Notificación, expresándose como una relación de composición. Tiene la responsabilidad de la creación de los objetos de la clase Notificación, así como las operaciones de modificación, búsqueda y eliminación de dichos objetos.

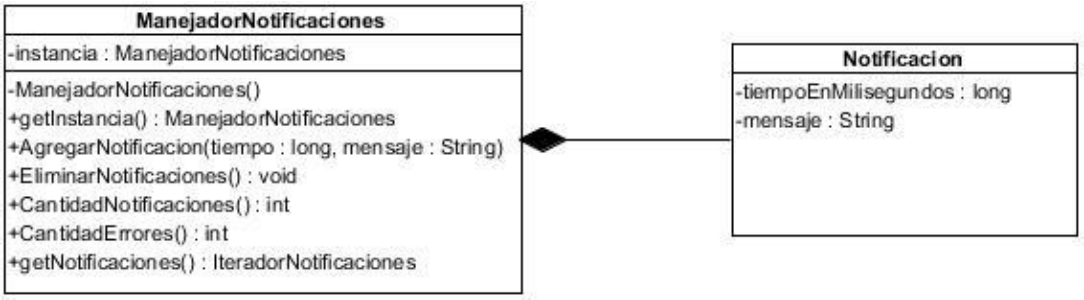

Figura 14 Uso del patrón creador en la solución

Fuente: elaboración propia

Para las clases que son creadas en un único momento y consultada posteriormente en todo el sistema fue empleado el patrón instancia única (*Singleton* en inglés). Dicho patrón permite que se creen objetos con una única instancia, controlando de igual forma el acceso al mismo. En la solución creada fue utilizado para la construcción del manejador de notificaciones de la capa de negocio, representado en la Figura 16.

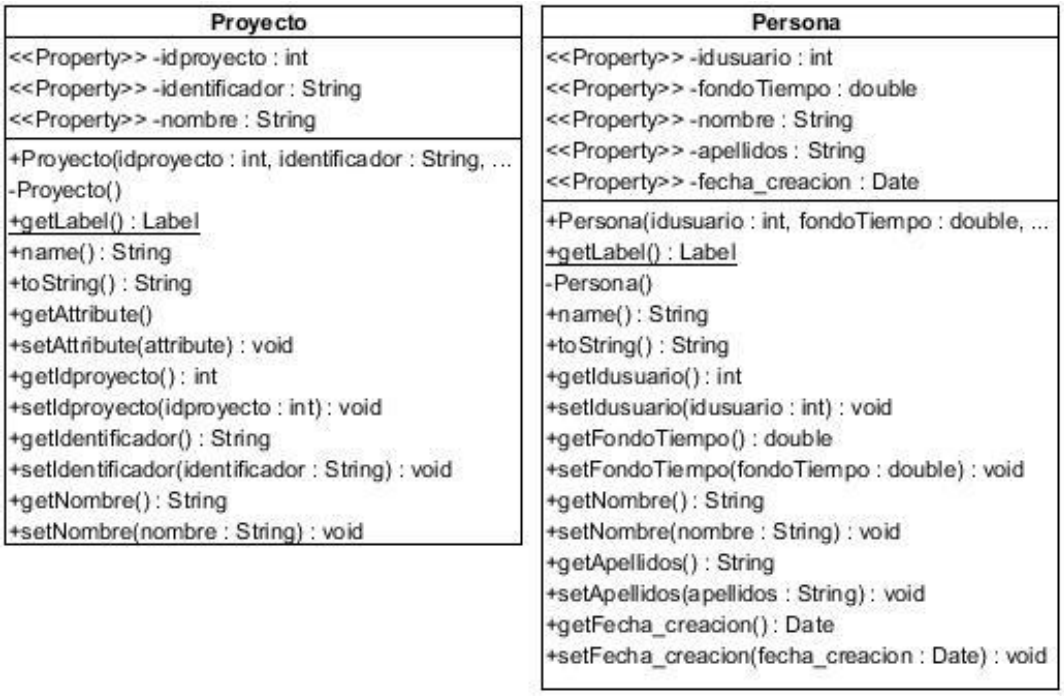

Figura 15 Uso del patrón experto en la solución

Fuente: elaboración propia

Esta posee su constructor privado, impidiendo que pueda ser instanciado en cualquier ubicación del código fuente. Para su implementación, y como define el patrón, se utiliza una instancia estática de dicho componente. Fue utilizado en la clase ManejadorNotificaciones encargada del manejo del fichero "errores.txt", que almacena los registros de eventos del sistema.

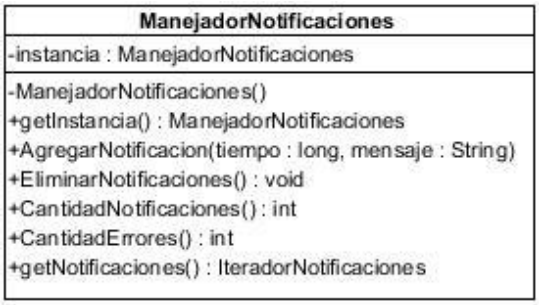

Figura 16 Uso del patrón instancia única en la solución

Fuente: elaboración propia

En cuanto a los objetos consultados desde la capa de acceso a datos, fue utilizado el patrón iterador (*Iterator* en inglés) para recorrer cada uno de los registros obtenidos. Su uso garantiza la integridad de la información, sin necesidad de revelar el contenido interno de los objetos listados. Como parte de la solución fue utilizado en múltiples escenarios, como en el trabajo con los elementos del grafo, acceso a datos en el modelo relacional, así como en el control de las diferentes estructuras de datos agrupadas.

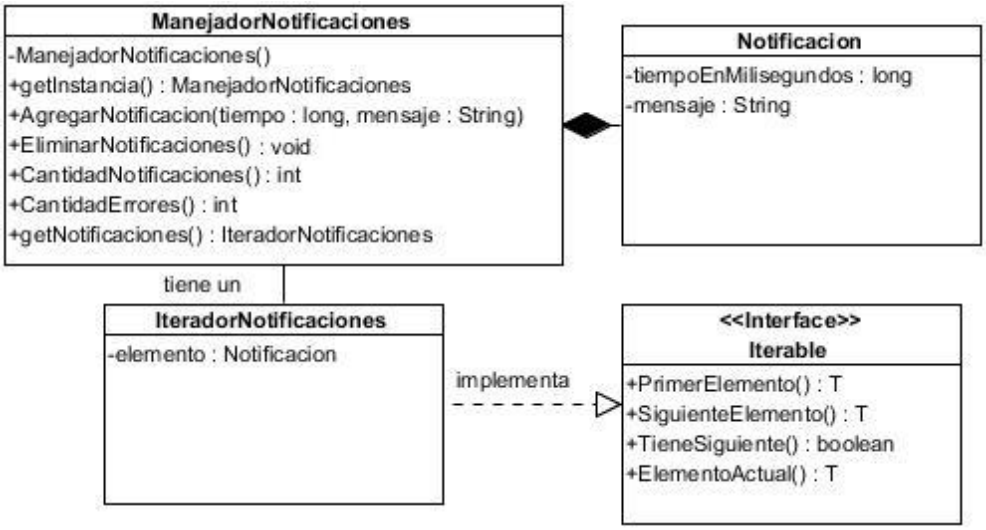

Figura 17 Uso del patrón Iterador en la solución

Fuente: elaboración propia

El patrón observador (*Observer* en inglés) fue empleado durante los procesos en segundo plano para la realización de las notificaciones. Evita de esta forma, acceder a hilos de tareas que puedan bloquear el sistema. En la herramienta desarrollada fue utilizado entre las clases GesproDBTransformer que comprende el proceso de extracción, transformación y carga de los datos del modelo relacional hacia el modelo orientado a grafo, y la clase Transformación, que notifica su estado en la interfaz correspondiente. En la Figura 18 puede apreciarse el diseño de dichas clases, así como el uso del patrón en ellas.

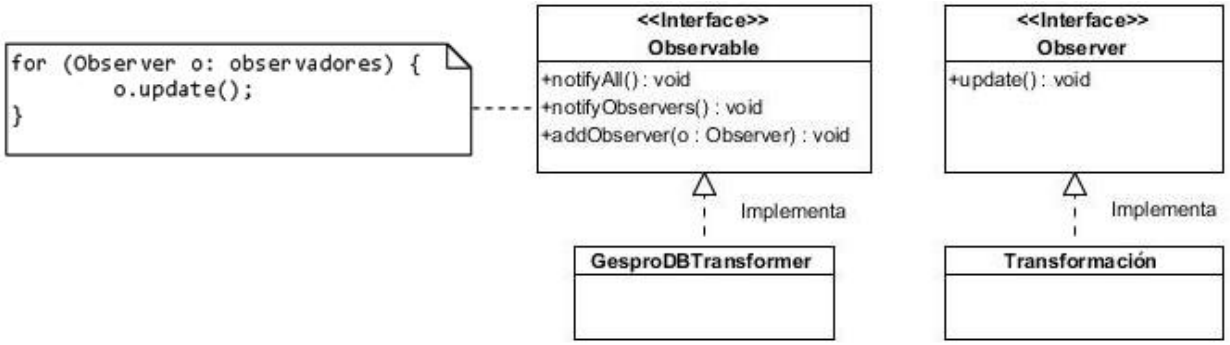

Figura 18 Uso del patrón observador en la solución

Fuente: elaboración propia

Tarjetas CRC

Las tarjetas CRC son propuestas en (Beck & Andres, Extreme Programming Explained: Embrace Change, 1999) con el objetivo representar los objetos involucrados en las historias de usuario conformadas, así como las responsabilidades que tienen en el escenario en que se encuentran. Para el presente trabajo se confeccionaron un total de cinco tarjetas CRC, expuestas en el Anexo 3.

- Conexión
- ManejadorPostgreSQL
- ManejadorNeo4j
- CalculoIndicadores
- ManejadorVistas

En la Tabla 13 se muestra la tarjeta CRC correspondiente a la clase Conexión, apreciándose su colaboración con las clases ManejadorPostgreSQL y ManejadorNeo4j.

Clase: Conexión		
Responsabilidades	Colaboradores	Métodos
• Establecer conexión con el sistema de base de datos relacional de Xedro-GESPRO. • Establecer conexión con la base de datos orientada a grafos.	• ManejadorPostgreSQL • ManejadorNeo4j	• Conectar
Observaciones: dados los parámetros de conexión, se comprueba la existencia de la base de datos, notificándose si se establece o no la conexión.		

Tabla 13 Tarjeta CRC para la clase Conexión

Conclusiones parciales

La realización del análisis sobre los indicadores expuestos en la sección 1.2, permitió identificar los diferentes elementos del modelo relacional que hicieron posible la confección del modelo orientado a grafo, contribuyendo a la comprensión de las relaciones que existen entre los conceptos asociados. De igual forma, propició el diseño de los algoritmos necesarios para la estimación de dichos indicadores a partir del modelo propuesto.

Las diferentes fases de la metodología guiaron el proceso de desarrollo de software, posibilitando la creación de los diferentes artefactos, la captura de los requisitos, la definición de las historias de usuario, así como las tarjetas CRC para definir el comportamiento y estructuración de las clases del sistema.

La definición de la arquitectura basada en capas, sus componentes y funcionalidades correspondientes permitió obtener una abstracción de la solución, siendo la base para el desarrollo de la misma. El uso de patrones de diseño contribuye a la reusabilidad, flexibilidad y mantenimiento de la herramienta desarrollada, disminuyendo la aparición de problemas frecuentes en el código generado, y su rápida solución una vez detectados.

CAPÍTULO 3: Validación de la solución

Introducción

Luego de culminadas las diferentes etapas de diseño y concepción de la herramienta propuesta como solución, se introduce en el presente capítulo el proceso de verificación de la calidad. Para ello, son expuestos los resultados de las pruebas realizadas al software, soportadas las pruebas unitarias que propone la metodología XP, las pruebas de aceptación en las que se involucra el cliente, así como un caso de estudio donde se exponen los resultados comparativos del cálculo de los indicadores en el sistema Xedro-GESPRO y la herramienta confeccionada.

3.1 Pruebas unitarias

Como parte del ciclo de vida del desarrollo del software, las pruebas forman parte indispensable para llegar a la culminación y entrega del producto realizado. La metodología seleccionada (XP), propone la realización de pruebas constantes para verificar el correcto funcionamiento de los diferentes módulos, componentes o partes del sistema, denominadas pruebas unitarias (Beck & Andres, Extreme Programming Explained: Embrace Change, 1999).

En el presente trabajo, fue utilizado dentro del nivel de pruebas unitarias o de componentes (Zapata Sánchez, 2013), el tipo de prueba funcional, y como método a emplear el de caja blanca. Para su realización fueron seleccionados los algoritmos diseñados para calcular los indicadores en el modelo propuesto, las clases de negocio que interactúan con los datos y que notifican los diferentes resultados.

Como herramienta para realizar las pruebas de forma automática, se utilizó el marco de trabajo JUnit en su versión 4.6, el cual es ampliamente utilizado para los sistemas desarrollados con el lenguaje Java. El entorno de desarrollo Netbeans 8.0, seleccionado como herramienta de desarrollo de la solución (ver sección 1.5), permite realizar las pruebas unitarias con JUnit en el propio contexto de desarrollo mediante la utilización de bibliotecas. Una vez planificadas cuáles son las clases indispensables a probar, se realiza una primera iteración sobre los diferentes escenarios del sistema (ver sección 2.3.2), obteniendo así resultados que se exponen en la Figura 19.

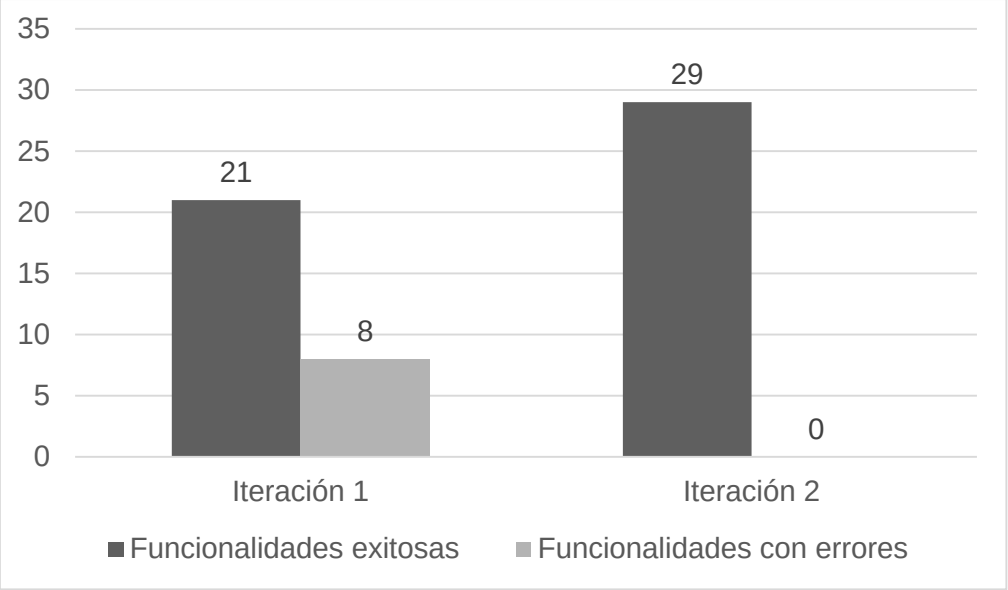

Figura 19 Resultados de las pruebas unitarias con JUnit

Fuente: elaboración propia

A raíz de los resultados de la primera iteración, el equipo de desarrollo realizó las acciones pertinentes para corregir los diferentes errores que impidieron culminar las pruebas con éxito. Se verificó cada prueba diseñada, y tras una segunda iteración, se dio cumplimiento al objetivo trazado, cumpliéndose a un cien por ciento todas las pruebas unitarias.

3.2 Pruebas de caja negra

Las pruebas de aceptación es un proceso en donde el cliente da su conformidad sobre el funcionamiento de los requisitos inicialmente pactados. Como principal requerimiento en esta etapa se encuentra la presencia del cliente en la elaboración de los casos de prueba a partir de las historias de usuario confeccionadas.

Para ello, se elaboraron veintitrés casos de prueba, respondiendo a todas las historias de usuario, y los requisitos asociados a cada una de ellas. En la Tabla 14 se detalla un ejemplo de los diferentes elementos que contienen cada uno de los casos de pruebas, como son los escenarios, flujos y respuesta del sistema. A partir de un caso de prueba, se define cada una de las variables de entrada.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1: parámetros de configuración con datos correctos	Se introducen los parámetros de conexión correctos y se acciona el botón “Probar conexión”	El sistema muestra el mensaje "La conexión fue establecida con éxito"	Se inicia la aplicación, se acciona el botón “Siguiente” y en la vista Inicio se selecciona “Crear nueva transformación” y se acciona el botón “Siguiente”
EC 1.2: parámetros de configuración con datos incorrectos	Se introducen los parámetros de conexión incorrectos y se acciona el botón “Probar conexión”	El sistema muestra el mensaje de error "Falló la conexión al servidor 10.8.8.10, verifique su conexión y que el sistema de base de datos esté funcionando"	Se inicia la aplicación, se acciona el botón “Siguiente” y en la vista Inicio se selecciona “Crear nueva transformación” y se acciona el botón “Siguiente”
EC 1.3: parámetros de configuración con datos incompletos	Se introducen los parámetros de conexión incompletos y se acciona el botón “Probar conexión”	El sistema muestra el mensaje de error "El campo Dirección IP de la BD Xedro-GESPRO está vacío"	Se inicia la aplicación, se acciona el botón “Siguiente” y en la vista Inicio se selecciona “Crear nueva transformación” y se acciona el botón “Siguiente”

Tabla 14 Caso de prueba “Verificar conexión”

Los valores de entrada para el desarrollo de los casos de prueba son definidos a través de las variables. Para cada caso de prueba fue elaborada una descripción de las variables que se emplean. En la Tabla 15, se puede observar el conjunto de variables definidas para el caso de prueba “Verificar conexión”. Como principales atributos recogidos en la misma se encuentra el nombre del campo a probar, su clasificación, si puede tener valor nulo, y una breve descripción.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Dirección IP	Campo de texto	Sí	Dirección IP de la base de datos del sistema Xedro-GESPRO
2	Puerto de conexión	Campo de texto	Sí	Puerto de conexión del sistema gestor de base de datos

3	Usuario	Campo de texto	Sí	Usuario para la autenticación con el sistema gestor de base de datos
4	Contraseña	Campo de texto	Sí	Contraseña para la autenticación
5	Nombre de la BD	Campo de texto	Sí	Nombre de la base de datos en el sistema gestor de base de datos

Tabla 15 Conjunto de variables para el caso de prueba "Verificar conexión"

Para la realización del caso de pruebas, al conjunto de variables definidas se les asigna un juego de datos para los diferentes escenarios. Para el caso de prueba “Verificar conexión”, se definió el siguiente conjunto de valores para las variables definidas (ver Tabla 16).

No	Nombre del campo	Escenario		
		1.1	1.2	1.3
1	Dirección IP	localhost	10.8.8.10	(Vacío)
2	Puerto de conexión	5432	5432	5432
3	Usuario	postgres	oracle	postgres
4	Contraseña	postgres	123	postgres
5	Nombre de la BD	gespro	gespro_BD	gespro_new

Tabla 16 Juego de datos de las variables en los diferentes escenarios

Una vez realizados todos los casos de prueba, se recogen las no conformidades detectadas por el cliente. En la Figura 20 se muestran los resultados de las pruebas de aceptación para las diferentes iteraciones, donde en la última, no se detectan señalamientos, cerrándose de esta forma el proceso de prueba descrito. Por último, el producto fue liberado por los especialistas en calidad del centro CEGEL perteneciente a la universidad (ver Anexo 4).

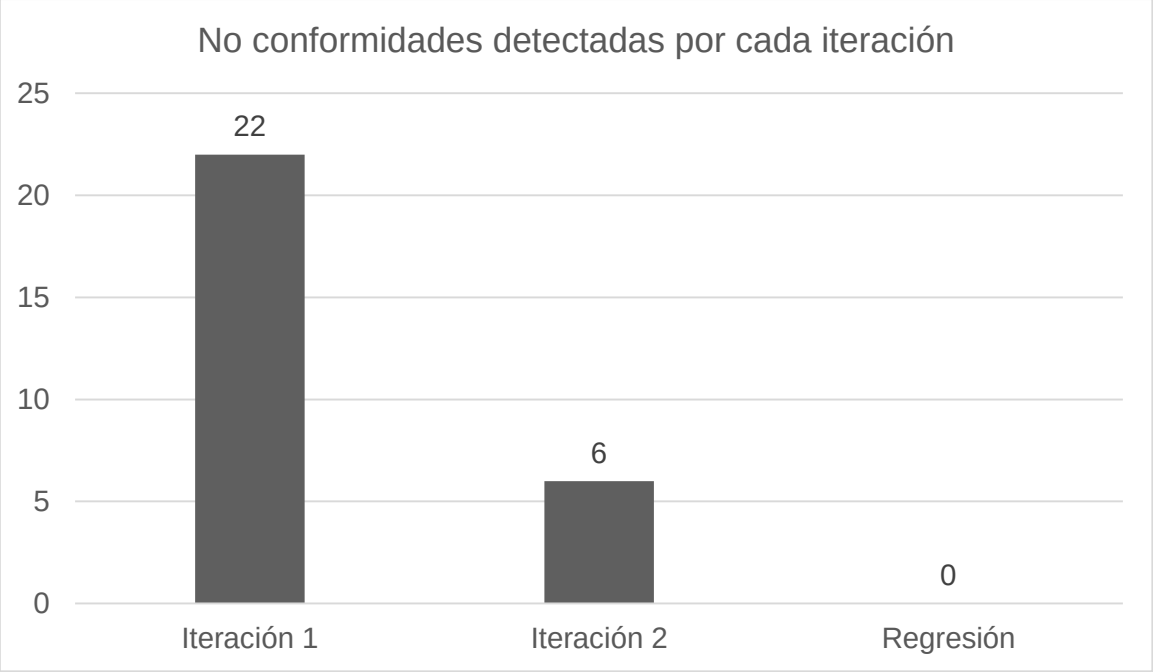

Figura 20 Resultados de las pruebas de aceptación

Fuente: elaboración propia

3.3 Caso de estudio

Las pruebas de comparación son experimentos que evalúan una o más herramientas para comparar su comportamiento. Para realizar las pruebas se deben definir métricas, que en el marco del estudio son los tiempos en que se calculan los indicadores.

Para la realización del experimento, se utilizaron los algoritmos para calcular los indicadores IRHT e IRHE, del sistema Xedro-GESPRO y de la herramienta confeccionada (a partir de ahora GESPRO-Graph), al igual que una fuente de datos de prueba común para ambos. Se realiza esta selección tomando a consideración de los especialistas del sistema Xedro-GESPRO, los cuales confirman son los indicadores que requieren mayor y menor tiempo respectivamente. Las pruebas son realizadas a nivel de persona, pues son la mínima unidad para el cálculo de estos indicadores. Para ello son necesarios dos parámetros: identificador de la persona y fecha de corte. Las prestaciones de la estación de trabajo utilizada son las siguientes:

- Microprocesador doble núcleo AMD E-450 a 1.6 GHz.

- Memoria RAM de 4GB.
- Almacenamiento físico (disco duro) de 500GB.

El experimento se ejecutó de la siguiente forma: se seleccionan diez casos de prueba, donde cada uno recoge el identificador de la persona y la fecha de corte. La selección se realizó de forma aleatoria, y puede apreciarse en la Tabla 17. Se calcularon en dos iteraciones los indicadores IRHT e IRHA en cada uno de los sistemas, tomándose como resultado final el promedio de los tiempos empleados para el cálculo.

Caso de prueba			Sistema Xedro – GESPRO		Sistema GESPRO – Graph	
No	Identificador	Fecha de corte	IRHT	IRHE	IRHT	IRHE
1	164	01/05/2015	478667	27	49	15
2	51	01/05/2015	465021	68	33	11
3	166	01/05/2015	511765	24	23	11
4	122	01/05/2015	487399	38	31	12
5	214	01/05/2015	462093	49	27	17
6	169	01/05/2015	464393	26	28	11
7	171	01/05/2015	485964	24	35	12
8	182	01/05/2015	490664	25	24	9
9	145	01/05/2015	502364	66	31	11
10	267	01/05/2015	424209	23	28	10

Tabla 17 Tiempos de respuesta en milisegundos del cálculo de los indicadores en ambos sistemas.

Luego de realizado el experimento, se obtienen los resultados que se muestran en la Figura 21. Como puede apreciarse, los tiempos de respuesta para ambos indicadores, son menores en el sistema desarrollado.

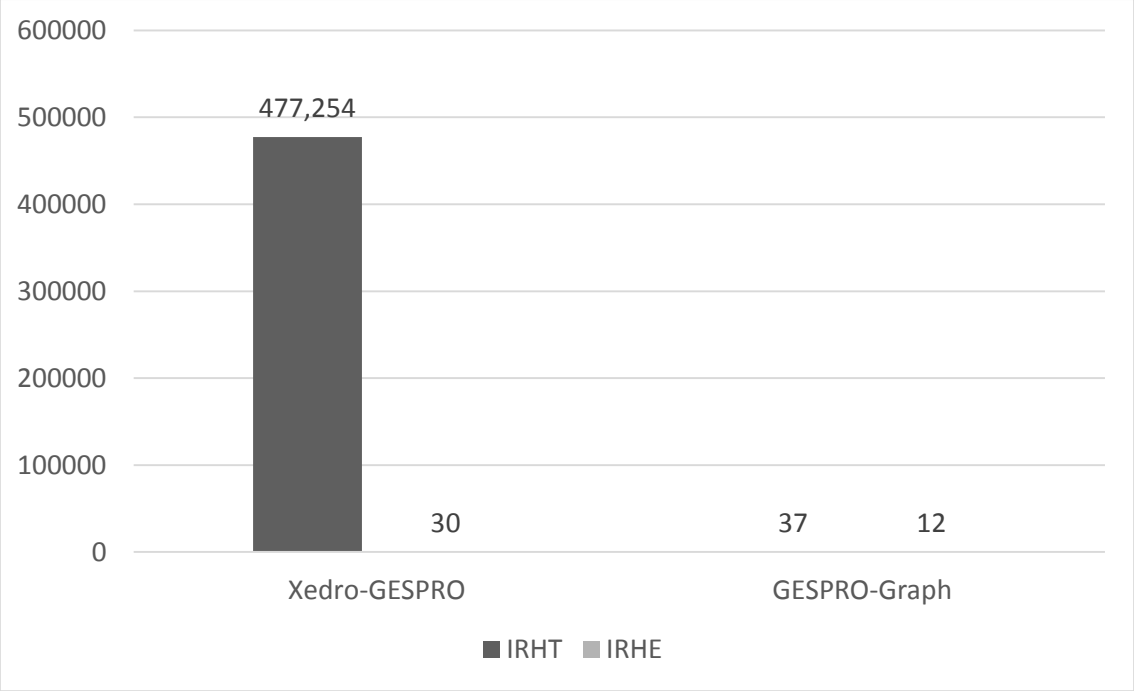

Figura 21 Resultados del tiempo de respuesta en milisegundos

Fuente: elaboración propia

Conclusiones parciales

En el presente capítulo se realizó la verificación de las funcionalidades y el cumplimiento de los requisitos especificados por el cliente, a través de las pruebas unitarias y el marco de pruebas JUnit. Para la validación de la solución se realizaron pruebas de caja negra, además de un caso de estudio como experimento demostrativo de la variable dependiente definida en el diseño metodológico. Por lo anterior, se llega a las siguientes conclusiones:

- La herramienta desarrollada, así como la correcta generación de un grafo donde se representan los nodos y relaciones entre los conceptos definidos en la Figura 2, validan el modelo propuesto.
- El cálculo de los indicadores requiere menor tiempo computacional utilizando el modelo propuesto, en comparación con el modelo relacional empleado por el sistema Xedro-GESPRO.
- La herramienta confeccionada dio cumplimiento a los requisitos definidos por el cliente.

CONCLUSIONES

La investigación realizada permitió la comprensión de la base conceptual para la construcción del modelo propuesto, la definición de una herramienta que lo soporte, así como la fundamentación de la metodología, las herramientas y tecnologías necesarias para el desarrollo de la solución.

El análisis de los indicadores relacionados con los recursos humanos en el sistema de Xedro – GESPRO, permitió identificar los elementos necesarios para conformar el modelo orientado a grafo y realizar el diseño de los algoritmos para el cálculo de los indicadores en el nuevo modelo.

La herramienta propuesta permitió a la transformación de los datos del sistema Xedro – GESPRO en el modelo confeccionado, y contribuyó al análisis de los indicadores relacionados a los recursos humanos, así como la representación de vistas para el apoyo a la toma de decisiones.

Con la realización de un caso de estudio, se validó la hipótesis propuesta, lográndose el cálculo de los indicadores relacionados a los recursos humanos de forma eficiente a través de la herramienta desarrollada.

RECOMENDACIONES

El presente trabajo, tiene como objeto de análisis los indicadores relacionados a los recursos humanos en el sistema de Xedro – GESPRO. A partir de que el modelo propuesto puede ser extensible, agregando nuevos conceptos y relaciones entre ellos, se realizan las siguientes recomendaciones:

- Extender el modelo propuesto para realizar el análisis de indicadores sobre los restantes indicadores definidos en (Lugo García, 2012).
- Definir un conjunto de vistas más amplio para el análisis de indicadores no estadísticos, utilizando para ello las propiedades que posee el modelo matemático grafo.

BIBLIOGRAFÍA

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction.* California: Oxford University Press.

Almuñez Mandryka, A., & Perez Santoyo, A. (2012). Análisis comparativo entre gestores de datos relacionales y gestores de datos NoSQL.

Andrés, C. (12 de junio de 2011). *Ciencia cibernética.* Recuperado el 31 de marzo de 2015, de Ciencia Cibernética: http://cienciacibernetica.blogspot.com/2011/06/teoria-complejidad-computacional-carlos.html

Apache.org. (8 de enero de 2015). *Commons Math: The Apache Commons Mathematics Library*. (Apache.org) Recuperado el 12 de febrero de 2015, de http://commons.apache.org/proper/commons-math/

Bass, L., Clements, P., & Kazman, R. (1998). *Software Architecture in Practice.* EEUU: Adison-Wesley.

Beck, K., & Andres, C. (1999). *Extreme Programming Explained: Embrace Change* (Vol. 2). EEUU: Addison-Wesley Professional.

Beck, K., & Cunningham, W. (16 de febrero de 2015). Obtenido de A Laboratory For Teaching Object-Oriented Thinking: http://c2.com/doc/oopsla89/paper.html

Camacho, E. (2010). NoSQL la evolución de las bases de datos. *I*(28).

Cárdenas-Haro, J. A., Vargas-Figueroa, A., & Maupome-Polanco, A. (2013). La notación asintótica en el cómputo científico. *Tlatemoani*.

Cytoscape. (3 de diciembre de 2014). *Cytoscape*. Obtenido de Network Data Integration, Analysis, and Visualization in a Box: http://www.cytoscape.org/what_is_cytoscape.html

D. Ullman, J., & Widom, J. (1999). *Introducción a los sistemas de bases de datos.* New Jersey, USA: Prentice Hall.

Delgado Victore, R. (2003). *La Dirección Integrada de Proyectos haciendo uso de las Nuevas Tecnologías de la Informática y las Comunicaciones.* La Habana: CETA ISPJAE.

Fischer, T., Niere, J., Torunski, L., & Zündorf, A. (2000). *Story Diagrams: A New Graph Rewrite Language Based on the Unified Modeling Language and Java.* Berlín: Springer.

Fowler, M., & Sccott, K. (1999). *UML Gota a Gota.*

Frank Codd, E. (1990). *The relational model for database management.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of reusable object-oriented software.* EEUU: Adison-Wesley.

Garlan, D., & Shaw, M. (1994). An introduction to software architecture. *Software Engineering Institute Technical Report*, 10-15.

Gephi.org. (3 de diciembre de 2014). *Gephi - The Open Graph Viz Platform*. Obtenido de https://gephi.github.io/

Gracia del Busto, H., & Yanes Enríquez, O. (septiembre de 2012). Bases de datos NoSQL. *Telem@tica*, 21-33.

Grupo PowerData. (9 de julio de 2013). *Procesos ETL: Definición, Características, Beneficios y Retos*. Obtenido de Power Data. Especialistas en Gestión de Datos: http://www.blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/312584/Procesos-ETL-Definici%C3%B3n-Caracter%C3%ADsticas-Beneficios-y-Retos

Gyssens, M., Paredaens, J., Van den Bussche, J., & Van Gucht, D. (1994). A graph-oriented object database model. (IEEE, Ed.) *Knowledge and Data Engineering, IEEE Transactions on, 6*(4), 572-584. Obtenido de http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=298174&isnumber=7391

H. Corman, T., E. Leiserson, C., L. Rivest, R., & Stein, C. (2009). *Introduction to Algorithms.* London, England: MIT Press.

Holzschuher, F., & Peinl, R. (2013). Performance of Graph Query Languages. *Institute for Information Systems*, 2-5.

Horowitz, J. (1967). *Critical Path Scheduling: Management Control Through CPM and PERT.* EEUU: Ronald Press Co.

ISO/IEC. (2000). *Information technology -Software product quality-.* ISO.

Larman, C. (1999). *UML y patrones.* México: Prentice Hall.

Larman, C. (2004). *UML y Patrones.* Canada: PRENTICE-HALL.

Llobet Azpitarte, R., Alonso Jordá, P., Devesa Llinares, J., Miedes De Elías, E., Torres Goterris, F., & Ruiz Fuertes, M. (2008). *Introducción a la Programación Orientada a Objetos con Java.* Valencia, España: Universidad Politécnica de Valencia.

López Zaldívar, L. Á. (2012). Propuesta de solución a la Vista de Arquitectura de Datos del Sistema GESPRO 12.05. 1-2.

Lugo García, J. A. (2012). Modelo para el control de la ejecución de proyectos basados en indicadores y lógica borrosa. 69.

Martinez Gil, F. A., & Martín Quetglás, G. (2003). *Introducción a la programación estructurada en C.* Valencia: Universidad de Valencia.

Neo4j. (25 de diciembre de 2010). *Neo4j*. Obtenido de Neo4j: http://neo4j.com/developer/java/

Object Management Group. (8 de enero de 1997). *Unified Modeling Language™ (UML®) Resource Page*. (Object Management Group) Recuperado el 13 de febrero de 2015, de Unified Modeling Language™ (UML®) Resource Page: Getting Started with UML: http://www.uml.org/

Oracle Corporation. (1 de diciembre de 2000). *NetBeans IDE website*. (Oracle Corporation) Recuperado el 12 de febrero de 2015, de NetBeans IDE: https://netbeans.org/

Oracle Corporation. (6 de junio de 2015). *Oracle Technology Network*. Obtenido de The Byte Code Verification Process: http://www.oracle.com/technetwork/java/security-136118.html

Project Management Institute. (2004). *Guía de los fundamentos de la gestión de proyectos.* Pensilvania, USA: Project Management Institute, Inc.

Robinson, I., Webber, J., & Eifrem, E. (2013). *Graph Databases.* Sebastopol, CA, USA: O'Reilly Media, Inc.

Rodríguez Puente, R., & Ril Valentin, E. (2014). Modelado de relaciones existentes en un equipo de proyecto.

Sanchez , E., Letelier Torres, P., & Canós Cerdá, J. (2003). Mejorando la gestión de historias de usuario en eXtreme Programming . *ISSI*, 6-8.

Shannon, P., Markiel, A., Ozier, O., S. Baliga, N., T. Wang, J., Ramage, D., . . . Ideker, T. (2003). Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. *Cold Spring Harbor Laboratory Press*, 2-3.

Sommerville, I. (2006). *Software engineering 8th Edition.* China: Pearson Education Limited.

SourceForge. (1 de agosto de 2003). *The JUNG Framework Development Team*. (SourceForge) Recuperado el 11 de febrero de 2015, de Java Universal Network/Graph Framework: http://jung.sourceforge.net/index.html

Stellman, A., & Greene, J. (2005). *Applied Software Project Management.* EEUU: O'Really Media.

Tinoco Gómez, O., Pedro Pablo, R., & Salas Bacalla, J. (2010). Criterios de selección de metodologías de desarrollo de software. *Industrial Data, 13*(2), 2-3.

Van Bruggen, R. (2014). *Learning Neo4j.* Birmingham, United Kingdom: Packt Publishing Ltd.

Visual Paradigm International. (8 de enero de 2004). *Visual Paradigm*. (Visual Paradigm International) Recuperado el 13 de febrero de 2015, de Visual Paradigm: http://www.visual-paradigm.com/

Wasserman, S., & Faust, K. (1994). *Social Network Analysis: Methods and Applications.* Illinois, EEUU: Cambridge University Press.

Zapata Sánchez, J. (21 de enero de 2013). *Pruebas del software: ingeniería de software con énfasis en pruebas.* Recuperado el 22 de marzo de 2015, de https://pruebasdelsoftware.wordpress.com/