Universidad de las Ciencias Informáticas Facultad 4

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor:

Ruber Rodriguez Dominguez

Tutores:

MSc. Roberto Millet Luaces

Título: Construcción semi-automática de secuencia de tareas de proyectos en función de requisitos o prerrequisitos.

Julio 2018

Frase

“Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el maravilloso mundo del saber.”

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de la presente tesis que tiene por título: “Construcción semi-automática de secuencia de tareas de proyectos en función de requisitos o prerrequisitos.” y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los 2 días del mes de junio del año 2018.

Biblioteca

_____

Firma del Autor

Roberto Millet Luaces

_____

Firma del Tutor

DATOS DE CONTACTO

Datos de Contactos.

Tutor: MSc. Roberto Millet Luaces.

Breve currículo:

• Profesor de Matemática.

• Graduado de Ingeniero Eléctrico en 1986, en Universidad de Camagüey.

• Profesor Auxiliar.

• MSc. en Ciencias Matemáticas.

• Imparte docencia en universidades desde 1987.

Ubicación: UCI, Cuba.

E-mail: milletp@uci.cu

AGRADECIMIENTOS

Le agradezco todo a toda mi familia, a mi tutor por su compromiso y por darme aliento cuando lo necesité, a la buenas amistades que he hecho en la UCI, principalmente a Osmany que me ayudó mucho en la tesis y es la persona en la que más he confiado en mis años de estudiante en la UCI, Anabel que siempre se preocupó por mí a pesar de mi pesadez, Eduardo mi compañero de clases, habitación, estudio y demás, un amigo con el que siempre pude contar, Lidia siempre estuvo para mí cuando la necesite, Josué, Pepe, y todos mis compañeros de aula con lo que tengo muy buena relación y si no fuera por ellos talvez no estaría hoy día en mi discusión de tesis.

Dayanis

Jessika

Rigoberto David García Mauri.

José Luis Bencomo Atanay

Lidiexy

DEDICATORIA

La tesis se la dedico a mi mamá, a mi papa, a mis hermanas Erlienis y Yannara y a mi abuela Nelsy, son las personas que más amo en el mundo y las que siempre han estado ahí para mí, apoyándome en todo, dándome todo su amor, mejores personas en el mundo no pueden haber, no pido más que eso.

RESUMEN

En la actualidad, con frecuencia, los proyectos de software experimentan retrasos en cuanto al tiempo definido para su completa ejecución, por consiguiente, se hace necesario una correcta manipulación y gestión de tiempo de ejecución del proyecto. En las empresas de software constantemente se trabaja en la búsqueda de nuevas y novedosas soluciones para mejorar su desempeño en el área, ejemplo de esto se puede observar en los proyectos de la Universidad de Ciencias Informáticas (UCI). Por lo ante expuesto, surge la necesidad de utilizar técnicas del Algoritmo Colonia de Hormiga (ACH) para permitir la optimización de la secuencia de tareas en función de los requisitos disponibles en proyectos de inversión.
En el trabajo se destaca y se hacen sugerencias a la simultaneidad de tareas compartiendo los recursos siempre que sea posible, ya que esto evidencia la eficiencia de la productividad. Se establece una relación entre requisitos de software y tareas, construyendo un grafo que constituye la base para la aplicación de ACH. Se logra el desarrollo de una aplicación que permita generar y optimizar la secuencia de tareas en función de un conjunto de requisitos de software en proyectos que trabajan de forma simultáneamente, cumpliéndose así el objetivo trazado en la investigación.

Palabras clave: optimización, proyectos simultáneos, requisitos, tareas.

SUMMARY

Currently, software projects often experience delays in the time defined for their full execution, therefore, it is necessary to properly manipulate and manage project execution time. Software companies are constantly working on new and innovative solutions to improve their performance in the area, an example of this can be seen in the projects of the University of Computer Science (UCI). Therefore, the need to use techniques of the Ant Colony Algorithm (ACH) arises to allow the optimization of the sequence of tasks according to the requirements available in investment projects.

In the work we highlight and make suggestions to the simultaneity of tasks sharing resources whenever possible, as this shows the efficiency of productivity. A relationship is established between software requirements and tasks, building a network that forms the basis for the application of ACH. The development of an application that allows to generate and optimize the sequence of tasks according to a set of software requirements in projects that work simultaneously, thus fulfilling the objective of the research.

Keywords: optimization, simultaneous projects, requirements, tasks.

Tabla de Contenidos

ÍNDICE FIGURAS

Índice de tablas

Introducción

A través de los años la sociedad ha estado inmersa en distintos cambios tecnológicos, los cuales constituyen el resultado de una constante necesidad de mejora y simplificación de procesos productivos, del ahorro de recursos y la búsqueda de una mayor eficiencia. El uso de las tecnologías aporta un gran éxito, además de capacidad para explicar, transformar y prácticamente controlar el mundo. La ciencia adquiriere cada vez más importancia en la medida en que proyecta utilidad para todas las personas en diferentes ámbitos sociales.

En la sociedad actual se evidencia el creciente adelanto científico-técnico, siendo las computadoras uno de los recursos más utilizados. Las nuevas tecnologías marchan en la misma dirección del desarrollo de los distintos países del mundo, desempeñan un papel importante en la labor de seguir innovando y así incrementan el control y comprensión del entorno, obteniendo una mejor calidad de vida. Ningún país puede alcanzar cierto desarrollo obviando el uso de estas, las cuales han revolucionado significativamente el incremento y utilización de un conjunto de conocimientos propios de la tecnología por parte de la humanidad. Este proceso permite verdaderos y significativos cambios en las personas, transformando su estado. El trabajo en proyectos de inversión está en función de contribuir al desarrollo de la sociedad, fundamentalmente si se aplican simultaneidad en el compartimiento de recursos.

Un proyecto múltiple parte del concepto de proyecto, entendiéndose por proyecto al conjunto de las actividades que desarrolla una persona o una entidad para alcanzar un determinado objetivo. Estas actividades se encuentran interrelacionadas y se desarrollan de manera coordinada. Lo más difícil en la construcción de un sistema de software es decidir precisamente qué construir. No existe tarea con mayor capacidad de lesionar al sistema, cuando se hace mal. Ninguna otra tarea es tan difícil de rectificar a posteriori F. P. Brooks, 2007.

En el año 2001, Victor Basili encontró cerca de 88 errores en una Especificación de Requisitos de Software (ERS) de 400 páginas para el proyecto “A-7E Operational Flight Program”. Esta ERS fue escrita por un grupo de expertos en especificación de requisitos. La NASA ha sufrido dos accidentes espectaculares cuyo origen se atribuye a problemas durante la definición de los requisitos. Los requisitos contienen demasiados errores. La mayoría de estos errores podrían ser detectados al principio. No detectarlos incrementa los costes (tiempo, dinero) de forma exponencial.

En la ejecución de los proyectos son varios los factores a tener en cuenta, fundamentalmente la secuencia en que se realizan las tareas en función de un conjunto de requisitos indispensables en las sesiones de trabajo. En todo proyecto la planificación de tareas constituye un aspecto organizativo que contribuye a un

eficiente desarrollo. Esto se logra mediante un diagrama de Gantt, en el cual se representa el tiempo en el eje horizontal y las distintas tareas es las que se divide el trabajo mediante las barras horizontales. Cada barra tiene un tiempo de inicio y un tiempo de finalización y puede realizarse de forma exclusiva o simultánea con otra tarea.

Existen distintos métodos y algoritmos que se utilizan en problemas de optimización. Uno de estos es el ACH, metodología inspirada en el comportamiento colectivo de las hormigas en búsqueda de sus alimentos. Es un algoritmo genético que proporciona una técnica de optimización adaptativa vinculada con el proceso natural de la evolución. Basado en meta-heurística por medio de parámetros estocásticos denominados feromonas y visibilidad.

En cada iteración del algoritmo se construye una solución al problema a través de un grafo, dato abstracto utilizado para la representación de la información, donde cada arista representa las diferentes opciones que tiene en ese instante la hormiga. Las aristas tienen información heurística con la medida de su preferencia para registrar un movimiento desde un nodo a otro adyacente. Contiene a su vez el rastro de feromona con el cual se mide la deseabilidad aprendida en el movimiento de un nodo a otro. Entre las aplicaciones más frecuentes se tienen las redes neuronales, inteligencia artificial, optimización de funciones numéricas, sistemas difusos, procesamiento de imágenes, control de sistemas, problema del viajero, enrutamiento de vehículos, líneas de producción de autos, etc.

En investigaciones realizadas en la UCI, consultas a expertos, se evidencia la no aplicación del análisis de prioridad de tareas relacionadas con los requisitos de software de los proyectos, encontrando como deficiente el trabajo de asignación de las tareas a los requisitos, ya que no existe una aplicación que permita automatizar este proceso que se realiza de manera manual.

En el Centro de Consultaría de Desarrollo de Arquitecturas Empresariales (CDAE) se continúa dando pasos en esa dirección.
En correspondencia con lo antes expuesto se define como **problema científico:** ¿Cómo obtener una secuencia lógica de las tareas en función de los requisitos de software en proyectos informáticos que trabajan simultáneamente?, por lo que se plantea como **objeto de estudio:** El método de optimización Algoritmo Colonia de Hormigas en la optimización de secuencia de tarea, enmarcado en el **campo de acción:** Aplicación del Algoritmo Colonia de Hormigas en la obtención de secuencias de tareas en proyectos de software.

Teniendo en cuenta el problema planteado, se define como **objetivo general:** Desarrollar una aplicación que permita generar y optimizar la secuencia de tareas en función de un conjunto de requisitos de software en proyectos que trabajan de forma simultáneamente.

Para dar cumplimiento al objetivo planteado, se proponen los siguientes **Objetivos específicos:**

- Elaborar el marco teórico relacionando los aspectos fundamentales presentes en la investigación.
- Obtener las tareas en función de los requisitos de software en el proceso de optimización de proyectos.
- Construir el grafo ponderado como base de la aplicación del Algoritmo Colonias de Hormigas.
- Aplicar Algoritmo Colonias de Hormigas en la optimización de secuencia de tareas en proyectos que trabajan de manera simultáneamente.
- Validar la solución a través de los métodos definidos en la investigación.

Métodos utilizados en la investigación

Métodos teóricos:

Analítico–Sintético: se utiliza para organizar el conocimiento obtenido de la información consultada sobre el tema en relación con conceptos, definiciones y terminologías que se utilizan en el análisis de los Algoritmos Colonia de Hormigas y su aplicación en la optimización de recursos en proyectos, para una mayor facilidad en la realización del estudio del arte.

Histórico–Lógico: se utiliza para enfatizar en datos históricos-evolutivos sobre los algoritmos a utilizar, empleados en otros experimentos y actualmente, enfocado en el estudio y análisis de la planificación de proyectos simultáneos y los Algoritmos Bioinspirados, en particular en el Algoritmo Colonia de Hormigas.

Inductivo-Deductivo: se utiliza para obtener reglas a partir datos iniciales para llegar a conclusiones y análisis de conceptos y definiciones y de la aplicación de los Algoritmos Bioinspirados.

Métodos empíricos:

Consulta de expertos: se utiliza para obtener el criterio de especialistas en el tema de la investigación realizada.

Encuesta: se utiliza con el objetivo consultar expertos para obtener información relevante que contribuyan a esclarecer el objetivo de la investigación.

Experimento: este conjunto de acciones constituye en gran medida la base de la investigación realizada para verificar la utilidad de la teoría de colonia de hormigas en la construcción de un cronograma de proyecto.

Capítulo 1: Conceptos y definiciones

El capítulo referencia los aspectos teóricos utilizados en la investigación. Se explican las terminologías, conceptos y definiciones de la misma que constituyen la base del conocimiento de este trabajo. Además, se hace hincapié en la herramienta de desarrollo, así como el lenguaje de programación con el cual se implementa la aplicación.

1.1 Proyecto

Se denomina proyecto al esfuerzo emprendido con un único objetivo, la creación de un producto o servicio único, lo que requiere de una buena planificación, donde se diseñe el modo en que se utilizan los recursos de la organización para alcanzar las metas planteadas. En este sentido, puede determinarse que todo proyecto tiene un principio y un final, recursos definidos y unos objetivos buscados**¡Error! No se encuentra el origen de la referencia.**.

1.2 Planificación de Proyectos

La planificación de proyectos forma parte de la gestión de proyectos, la cual se auxilia de cronogramas tales como diagramas de Gantt para planear y subsecuentemente informar del progreso dentro del entorno del proyecto. Es el proceso para cuantificar el tiempo y recursos que un proyecto debe constar. La finalidad del planeamiento de proyecto es crear un plan de proyecto que un gestor de proyectos pueda usar para acompañar el progreso de su equipo.

Cuando se realiza la planeación de un proyecto es necesario tener en cuenta los puntos de convergencia de los proyectos. Para saber dónde están compartidos los recursos y los datos, donde están las fortalezas y las debilidades, ventajas y desventajas, entre otras cuestiones.

Por medio de la planeación el hombre moderno se propone resolver problemas complejos y orientar procesos de cambio, enfrentando múltiples desafíos, haciendo un amplio uso de los recursos que le proporcionan la ciencia, la técnica y la cultura.

Planificar un proyecto es diseñar acciones encaminadas a dar cumplimiento de determinados propósitos, buscando la mejor manera de utilizar racionalmente los recursos disponibles. Aplicado al mundo laboral, planificar y gestionar proyectos consiste en definir objetivos productivos de corto, mediano y largo plazo en función de los cuales se programan acciones y se ordenan tareas, bajo un régimen de control de gestión y evaluación de resultados**¡Error! No se encuentra el origen de la referencia.**.

1.3 Múltiples proyectos

El trabajo con proyectos múltiples está encaminado a administrar diferentes proyectos, reaccionar con eficiencia ante tareas en los procesos de desarrollo de productos, compartir entregables, recursos, información y tecnología, recolectar datos por diferentes integrantes del equipo y permitir la realización de distintos análisis con los datos recolectados. Su propósito es realizar comparaciones que arrojen resultados que ayuden en la toma de decisiones que tributen al trabajo eficiente.

En el marco de proyectos simultáneos, el trabajo de diploma se apoya en el concepto a través de cual se entiende como proyectos simultáneos a la ejecución de dos o más proyectos que comparten recursos o presentan de manera coordinada una dependencia en cuanto a la ejecución de una o varias tareas en las cuales comparten dichos recursos.

1.4 Ingeniería de Requisitos

La Ingeniería de Requisitos (IR) proporciona el mecanismo apropiado para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación, y administrar los requisitos conforme; estos se convierten en un sistema operacional. Establece procesos útiles y sistemáticos para llevarla a cabo, de manera que guía el desarrollo correcto hacia el sistema. Su importancia se sustenta al permitir una definición clara, consistente y compacta de las correctas especificaciones de requisitos que definen el comportamiento del sistema con el fin de minimizar al máximo los problemas presentados en el desarrollo del software y que tanto afectan al producto final.

La IEEE (del inglés *Institute of Electrical and Electronics Engineers*) publicó que un requisito es: “una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal” (IEEE 2006).

1.5 Importancia de la Ingeniería de Requisitos

Para una correcta elaboración de un software es de suma importancia tener un conocimiento pleno y claro sobre los diferentes requisitos del mismo que dan cumplimiento a las funcionalidades del software. De esta manera, el equipo de desarrollo tiene una visión exacta sobre lo que va a construir y para que el cliente quede en su total satisfacción y la calidad del producto no esté afectada en el momento de la entrega. Se observa con frecuencia que los usuarios no tienen la idea exacta de lo que se desea, cuales necesidades se pueden convertir en software y cuáles no. Debido a esto en la industria de software se le da solución a este problema con la definición de intermediarios entre los usuarios y los desarrolladores.

Estos intermediarios son los analistas y tienen como objetivo principal obtener los requisitos de cada usuario y de esta forma lograr una única visión, tanto del equipo de desarrollo como de los usuarios y componer una especificación de requisitos completa, correcta y consistente. Por estos motivos entre otros, en busca de mejoras en la rama, se crea la IR, reconocida desde 1990 y hasta la fecha. Variadas son las técnicas y herramientas que se han desarrollado y que han permitido aplicar esta disciplina no solo al ámbito del desarrollo de software, sino también a otras esferas.

1.6 Tareas

Las tareas pueden ser diversas, tales como, operaciones de un proceso de producción, despegues y aterrizajes en un aeropuerto. Cada tarea puede tener diferentes niveles de prioridad, tiempo de posible inicio y fin, disponibilidad para realizarlas, costo, etc. Los objetivos a perseguir pueden tomar varias formas: minimizar los tiempos de finalización de la última tarea; minimizar el número de tareas luego de una fecha de entrega acordada; minimizar las tareas asignadas simultáneamente.

1.7 Planificación de tareas

En todo proyecto la planificación de tareas constituye un aspecto organizativo que contribuye a un eficiente desarrollo. Esto se logra mediante un diagrama de Gantt. En dicho diagrama se representa el tiempo en el eje horizontal y las distintas tareas es las que se divide el trabajo se representan mediante barras horizontales. Cada barra tiene un tiempo de inicio y un tiempo de finalización y puede realizarse de forma exclusiva o de forma simultánea con otra tarea.

1.8 Grafo

Un grafo G es un par G = (V, A) donde V es un conjunto finito de elementos que se denominan vértices y A es un conjunto de pares no ordenados <x, y>, donde y∈ V y x∈ V, denominados aristas o arcos. Cuando se dice que A es un conjunto de pares no ordenados <x, y>, significa que los pares <x, y> y <y, x> se refiere a una misma arista, la que conecta los vértices x e y.
Un grafo ponderado G es un par G = (V, A) donde V es un conjunto finito de elementos que se denominan vértices, A es un conjunto de pares no ordenados <x, y>, donde (x∈ V) ^ (y∈ V), denominados aristas o arcos, donde a cada elemento de A (a cada arista) se le asocia un valor real positivo.

El peso asociado a cada arista puede interpretarse como el costo de trasladarse de un vértice a otro. Dicho costo puede ser, por ejemplo: el costo en tiempo, distancia, costo en algún determinado recurso. Esto posibilita que los grafos ponderados puedan ser aplicados en múltiples problemas.

Un grafo dirigido G es un par G = (V, A) donde V es un conjunto finito de elementos que se denominan vértices y A es un conjunto de pares ordenados <x, y>, donde (x∈ V) ^ (y∈ V), denominados aristas o arcos dirigidos.

Un grafo no dirigido es aquel en el que las conexiones no tienen una dirección establecida, es decir, una conexión entre dos vértices está definida en los dos sentidos posibles.

Algunas terminologías de la teoría de grafo necesarias para el presente trabajo son las de vértices adyacentes, camino y camino simple.

Vértices adyacentes: En un grafo G = (V, A), un vértice y es adyacente a otro vértice x si el par <x, y> es una arista del grafo G.
Camino: En un grafo G = (V, A) un camino de longitud *n* (*n* ≥ 0) es una sucesión de vértices v_0, v_1,…, v_n donde cada vértice v_k es adyacente v_k-1 para 1 ≤ *k* ≤ *n*. En este caso se dice que el camino va de v_0 a v_n.
Camino Simple: Un camino es simple si los vértices que lo componen son distintos excepto posiblemente el primero y el último.

1.9 Algoritmos Colonia de Hormigas

El Algoritmos Colonia de Hormigas (ACH) es una técnica probabilística para dar solución a problemas computacionales que pueden reducirse a buscar los mejores caminos o rutas en grafos. Este algoritmo fue propuesto inicialmente por marco Dorigo en 1992 en su tesis de doctorado, esta idea original se ha diversificado para resolver una amplia clase de problemas numéricos, y como resultado, han surgido una cantidad de problemas nuevos basándose en el comportamiento de las hormigas.

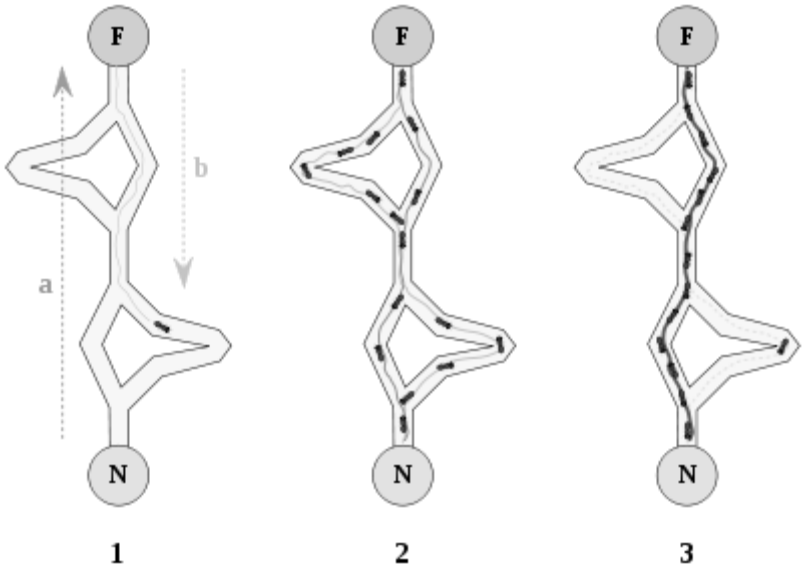

Fig. 1 Hormigas desde el nido a la fuente de alimento.

Las hormigas son prácticamente ciegas y por medio de movimientos al azar acaban por encontrar el camino más corto desde su nido hasta la fuente de alimentos, lo que no es posible por una sola hormiga, sino que es resultado de la colonia en conjunto.

En la imagen se representa el proceso de búsqueda de alimentos de las hormigas. En la representación identificada con el número 1, la hormiga transita aleatoriamente en busca de alimentos hasta que llega a donde está la fuente, luego de regreso a la colonia en forma más o menos directa, deposita una sustancia llamada feromona. Esta sustancia es atractiva para otras hormigas que siguen el rastro hasta encontrar la fuente de alimentación.

En la representación identificada con el número 2, las hormigas llegan al hormiguero por varios posibles caminos, depositando mayor cantidad de feromonas en el camino más corto debido a que permite mayor cantidad de recorridos y aumenta la atracción hacia esta ruta. Los caminos largos son cada vez menos visitados porque la feromona allí depositada va desapareciendo.

En la representación identificada con el número 3, las hormigas en conjunto pueden encontrar la ruta más corta desde el nido hasta la fuente de alimentación utilizando la feromona depositada por el hormiguero y su característica de evaporación que les posibilita alejarse de las rutas donde la sustancia tenga menos fortaleza y escoger el camino más corto y más utilizado por la colonia.

Los ACH se utilizan fundamentalmente en la solución de problemas de optimización combinatoria, problemas estocásticos, programación paralela y multiobjetivo y en la búsqueda de soluciones bastante

cercanas a las soluciones óptimas al problema del viajero. Cuando el grafo puede cambiar dinámicamente, el algoritmo puede seguir operando y adaptar los cambios en tiempo real muy aplicado en el enrutamiento de redes y en los sistemas de transporte urbanos.

La representación de la información para resolver la situación se hace por medio de una estructura de dato conocida por grafo. Este es una importante estructura de datos abstracta que tiene gran utilidad tanto en la informática como en la matemática.

1.10 Lenguaje de programación

Los lenguajes de programación constituyen un conjunto de reglas, herramientas y condiciones que permiten crear programas o aplicaciones dentro de una computadora. Estos programas son los que permitirán ordenar distintas acciones a la computadora en un idioma comprensible por ella. Como su nombre lo indica, un lenguaje tiene su parte sintáctica y su parte semántica. ¿Qué quiere decir esto? Que todo lenguaje de programación posee reglas acerca de cómo se deben escribir las sentencias y de qué forma. A su vez, los lenguajes de programación se dividen en tres grandes grupos: los lenguajes de máquina, los de bajo nivel y los de alto nivel.

Entre los lenguajes más populares se encuentran Java, C y C++, en ese orden de prioridad.
El lenguaje C++, es un lenguaje robusto, de buen manejo de la memoria, utiliza punteros, es orientado a objetos y tiene la ventaja de permitir sobre carga de operadores.

1.11 Herramienta de desarrollo

Un entorno de desarrollo integrado, es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto.

Ya dicho el lenguaje a utilizar, se presenta el entorno de desarrollo QT Creator en su versión 5.7.0, QT fue creado por Trolltech para el desarrollo de aplicaciones con las bibliotecas Qt.

Está disponible para los sistemas operativos Linux, Max y Windows, permitiendo al desarrollador crear aplicaciones para múltiples sistemas o plataformas móviles**¡Error! No se encuentra el origen de la referencia.**.

Capítulo 2: Obtención del conocimiento

Introducción

En este capítulo se aplican los conceptos y definiciones plasmados en el capítulo anterior con la finalidad de encontrar métodos y variantes que permitan construir el grafo ponderado que relaciona las tareas que dan cumplimiento a los requisitos del proyecto, dando parcial cumplimiento al objetivo planteado en la investigación.

2.1. Análisis estadístico de la base de datos obtenida por respuestas de los expertos

Luego de un conjunto de entrevistas a expertos en el tema de investigación se realiza una selección de las tareas a ejecutar de forma general en los proyectos, a las cuales en dependencia de sus criterios se le da una prioridad en cuanto a su importancia. Una vez determinado el nivel de importancia, se procede a modelar y ejemplificar con estas de mayor índice de importancia. El índice de más importancia es 1, lo que se resume en el acápite siguiente.

2.2. Análisis de las Tareas

En la Tabla 1 se muestra el resultado de la evaluación de los criterios de 18 especialistas de la UCI, en función del orden de prioridad de las tareas dentro de su clasificación, con el propósito de encontrar la media y la varianza de estos resultados, observándose la medida de dispersión de los datos alrededor de la media aritmética. Los datos obtenidos son muy próximos a 1 lo que evidencia una correcta dispersión de los datos y en alguna medida concordancia entre las respuestas dadas por los expertos.

Cuestionario realizado a especialistas.

En la tabla se muestra los criterios dados por los especialistas en cuanto a la prioridad de las tareas, a lo que se calcula la media aritmética para así encontrar la prioridad y la varianza para encontrar el nivel de incertidumbre de los criterios.

Tabla. 1 Obtención de las prioridades en tarea.

Clasificación	Tareas	Orden de prioridad	Promedio	Varianza
Estudio de viabilidad	Análisis del presupuesto	2,3,4,1,2,2,4,3,2,2,2,2,2,3,1,2	2 (2,3)	0,76
	Descripción del proyecto	6,6,6,8,7,7,7,5,7,7,7,7,7,7,6,7	7 (6,7)	0,50
	Definir y documentar posibles sistemas	7,5,5,7,7,6,6,6,5,7,7,5,5,5,6,5	6 (5,9)	0,78
	Analizar el coste de sistemas similares	2,2,1,1,1,1,1,2,1,1,1,1,1,2,1,1	1 (1,3)	0,21

	Estimar el tamaño del sistema.	3,3,3,2,2,2,3,3,3,2,3,3,3,3,2,3	3 (2,7)	0,23
	Estimar la planeación y los costos.	4,3,3,3,4,4,4,4,4,3,4,4,4,4,3,5	4 (3,8)	0,33
	Asignar un director de proyecto.	4,5,5,5,3,3,6,6,4,5,6,5,4,4,5,6	5 (4,8)	0,47
Análisis	Definir el ámbito del sistema propuesto.	6,6,3,3,3,5,4,4,4,4,4,5,3,4,5,3	4 (4,3)	1,00
	Consultas con usuarios	3,3,3,3,3,3,2,4,1,5,3,3,3,3,3,5	3 (3,1)	0,92
	Producir documento de requisitos	6,6,6,6,5,6,5,4,7,4,6,5,6,6,6,7	6 (5,7)	0,76
	Producir lista de beneficios tangibles e intangibles	6,7,6,7,7,6,7,7,7,6,6,7,7,7,6,7	7 (6,6)	0,25
	Construir prototipos.	1,2,2,3,3,1,1,3,3,2,2,2,2,3,2,1	2 (2,1)	0,55
	Definir requisitos.	2,1,2,1,1,1,1,3,2,2,1,1,1,1,1,1	1 (1,4)	0,38
	Especificación del sistema	4,5,6,4,5,5,5,4,6,5,5,5,4,4,6,5	5 (4,9)	0,52
Diseño	Producir el diseño global del sistema.	1,1,1,2,1,2,1,2,3,2,1,1,1,1,1,1	1 (1,4)	0,38
	Localización de paquetes de software.	5,3,3,4,4,4,4,5,4,4,4,3,3,5,4,4	4 (3,9)	0,46
	Desarrollar un diseño detallado del sistema, por alternativa de diseño planteada.	6,6,6,5,5,5,5,4,5,6,6,6,7,6,5,6	6 (5,6)	0,53
	Desarrollar un plan de prueba para el sistema.	2,2,4,3,1,4,3,1,2,2,2,4,2,2,2,3	2 (2,4)	0,93
	Desarrollar un plan de prueba diferenciado para cada alternativa.	2,3,3,3,2,4,3,3,3,2,2,4,2,3,3,4	3 (2,9)	0,52
	Identificar las necesidades de entrenamiento y documentación de los usuarios.	6,6,6,3,3,3,4,5,5,5,6,4,3,5,5,5	5 (4,6)	1,31
	Producir el documento de diseño y realizar una revisión final del documento de diseño del sistema.	6,7,6,7,7,7,5,7,5,6,7,7,7,6,7,7	7 (6,5)	0,53
Implementación	Plan de trabajo.	1,1,1,3,1,1,2,1,1,1,1,3,1,1,2,1	1 (1,4)	0,52
	Diseño de los programas (subsistemas).	1,1,1,2,2,2,3,2,3,1,1,3,1,1,2,4	2 (1,9)	0,92
	Codificar.	5,4,4,3,4,3,3,3,1,1,3,1,3,2,3,2	3 (2,8)	1,21
	Documentar cada programa.	4,3,5,4,4,3,4,3,3,6,4,4,4,3,3,5	4 (3,9)	0,78
	Probar cada programa.	3,6,4,4,4,4,5,5,4,5,6,3,7,5,5,5	5 (4,7)	0,33
	Prueba integral.	4,6,6,6,5,5,5,4,7,5,6,6,6,6,7,5	6 (5,6)	0,33
	Estimar costo, recursos y planificación de la etapa de prueba	7,6,6,7,7,6,7,7,5,6,7,7,7,6,7,7	7 (6,6)	0,40
Pruebas	Prueba del sistema.	1,1,1,1,1,3,1,1,2,2,1,1,1,1,1,1	1 (1,3)	0,33
	Documentar las pruebas	2,1,2,2,1,1,1,1,1,2,1,2,3,2,3,2	2 (1,6)	0,50
	Evaluar los documentos de entrega.	3,3,3,3,3,4,2,3,3,4,5,2,3,3,3,3	3 (3,1)	0,52

	Aprobar los documentos de entrega.	4,4,4,4,3,3,3,3,5,4,3,3,3,4,3,4	4 (3,6)	0,40
	Finalización del sistema completamente probado.	5,5,5,4,6,6,4,3,4,5,5,5,5,6,4,5	5 (4,8)	0,70

Clasificación (C), Tareas (T) y **Prioridad (P)**.

En la tabla se muestran las tareas organizadas por su clasificación y las prioridades obtenidas.

Tabla. 2 : Tareas con sus prioridades.

C	**T**	**P**
1. Estudio de viabilidad	1.1 . Análisis del presupuesto	2
	1.2 . Descripción del proyecto	7
	1.3 . Definir y documentar posibles sistemas	6
	1.4. Analizar el coste de sistemas similares	1
	1.5. Estimar el tamaño del sistema.	3
	1.6. Estimar la planeación y los costos.	4
	1.7. Asignar un director de proyecto.	5
2. Análisis	2.1 Definir el ámbito del sistema propuesto.	4
	2.2 Consultas con usuarios	3
	2.3 Producir documento de requisitos	6
	2.4 Producir lista de beneficios tangibles e intangibles	7
	2.5 Construir prototipos.	2
	2.6 Definir requisitos.	1
	2.7 Especificación del sistema	5
3. Diseño	3.1 . Producir el diseño global del sistema.	1
	3.2 . Localización de paquetes de software.	4
	3.3 . Desarrollar un diseño detallado del sistema, por alternativa de diseño planteada.	6
	3.4 . Desarrollar un plan de prueba para el sistema.	2
	3.5 . Desarrollar un plan de prueba diferenciado para cada alternativa.	3
	3.6 . Identificar las necesidades de entrenamiento y documentación de los usuarios.	5
	3.7 . Producir el documento de diseño y realizar una revisión final del documento de diseño del sistema.	7
4. Implementación	4.1 . Plan de trabajo.	1
	4.2 . Diseño de los programas (subsistemas).	2
	4.3 . Codificar.	3
	4.4 . Documentar cada programa.	4
	4.5 . Probar cada programa.	5

	4.6 . Prueba integral.	6
	4.7 . Estimar costo, recursos y planificación de la etapa de prueba	7
5. Pruebas	5.1 . Prueba del sistema.	1
	5.2 . Documentar las pruebas	2
	5.3 . Evaluar los documentos de entrega.	3
	5.4 . Aprobar los documentos de entrega.	4
	5.5 . Finalización del sistema completamente probado.	5

2.2.1 Obtención de variables por tareas.

La Tabla 3 muestra la relación de tareas con sus variables definidas como: T_1, …, T_N.

Tabla. 3 Variables de las tareas.

T	**ST**	**Variables**
Estudio de viabilidad	Analizar el coste de sistemas similares	T1
	Analizar el presupuesto	T2
	Estimar el tamaño del sistema.	T3
	Estimar la planeación y los costos.	T4
	Asignar un director de proyecto.	T5
Análisis	Definir requisitos	T6
	Construir prototipos.	T7
	Consultas con usuarios	T8
	Definir el ámbito del sistema propuesto.	T9
Diseño	Producir el diseño global del sistema.	T10
	Desarrollar un plan de prueba para el sistema.	T11
	Desarrollar un plan de prueba diferenciado para cada alternativa.	T12
	Localización de paquetes de software.	T13
Codificar	Plan de trabajo.	T14
	Diseño de los programas (subsistemas).	T15
	Codificar.	T16
	Documentar cada programa.	T17
	Probar cada programa.	T18
	Prueba integral.	T19
Prueba	Prueba del sistema	T20
	Documentar las pruebas	T21
	Evaluar los documentos de entrega	T22

	Aprobar los documentos de entrega	T23
	Finalización del sistema completamente probado.	T24

2.3.Especificaciones de requisitos

Las especificaciones de requisitos son consideradas como el producto de trabajo de la ingeniería de requisitos, basado en la colección, organización y representación de todos los requisitos del proyecto. Su importancia está representada en la posibilidad de establecer las bases de un acuerdo entre clientes/usuarios y equipo de desarrollo sobre los que el producto de software debe hacer, reducir el esfuerzo para evitar rediseños, recodificación y repruebas. Además, provee las bases para la estimación del costo y entrega del producto. Es indudable la influencia de la calidad de los requisitos y sus especificaciones en la calidad de la solución final. Muchos son los ingenieros de software que se han visto forzados a lidiar con especificaciones incompletas, inconsistentes o ambiguas y han experimentado la confusión que invariablemente provocan, reflejándose los resultados en la fecha de entrega y la calidad del software. Algunos autores han definido formalmente atributos para medir la calidad de tales especificaciones de requisitos.

2.4. Análisis de los Requisitos

La descripción detallada prueba que los requisitos una vez implementados ayudan considerablemente a su verificación. Los requisitos utilizados en la investigación forman parte de una base de datos de requisitos del CDAE, de una población de 52 requisitos se tomó una muestra de 32, para un 62% del total.

R1: Filtrar planificación de calidad

Descripción:

Cuando el usuario acceda a la ruta: Planificación/Alcance y Calidad/Planificación de Calidad (a nivel organizacional), el sistema debe permitir filtrar por proyecto las planificaciones de calidad correspondientes.

R2: Filtrar no conformidades de pruebas

Descripción:

Cuando el usuario acceda a la ruta: Planificación/Alcance y Calidad/NC Pruebas (a nivel organizacional), el sistema debe brindar la opción de filtrado “Filtros” mediante la cual debe permitir al usuario insertar los datos.

R3: Listar no conformidades de revisión

Descripción:

Cuando el usuario acceda a la ruta: Planificación/Alcance y Calidad/NC Revisiones (a nivel organizacional), el sistema deber listar todas las no conformidades que se tienen almacenas en la base de datos.

R4: Filtrar no conformidades de revisión

Descripción:

Cuando el usuario acceda a la ruta: Planificación/Alcance y Calidad/NC Revisiones (a nivel organizacional), el sistema debe brindar la opción de filtrado “Filtros” mediante la cual debe permitir al usuario insertar los datos.

R5: Adicionar contrato con clientes

Descripción:

Para adicionar un nuevo contrato con clientes se debe acceder a nivel organizacional al Módulo Contratación. Luego se debe seleccionar la etiqueta con el icono de “Nuevo contrato” para que se muestren los campos a llenar para el nuevo contrato. Una vez que se hayan introducido los datos se debe presionar el botón Aceptar y se muestra automáticamente el nuevo contrato en la tabla contratos con clientes.

R6: Editar contrato con clientes

Descripción:

Para editar un contrato con clientes se debe acceder a nivel organizacional al Módulo Contratación. Luego se debe seleccionar la etiqueta con el icono del “editar” en el contrato que se desee editar para que se muestren los campos a llenar. Una vez que se hayan introducido los datos se debe presionar el botón Aceptar y se muestra automáticamente el contrato editado en la tabla contratos con clientes.

R7: Eliminar contrato con clientes

Descripción:

Para eliminar un contrato con clientes se debe acceder a nivel organizacional al Módulo Contratación. Luego en el contrato que se desee eliminar se debe seleccionar la etiqueta con el icono “eliminar”.

R8: Copiar contrato con clientes

Descripción:

Para copiar un contrato con clientes se debe acceder a nivel organizacional al Módulo Contratación. Luego se debe seleccionar la etiqueta con el icono de “copiar” en el contrato que se desee copiar para que se muestren los campos a llenar.

R9: Adicionar contrato con proveedores

Descripción:

Para adicionar un nuevo contrato con Proveedores se debe acceder a nivel organizacional al Módulo Contratación. Luego se debe seleccionar la pestaña Contratos con Proveedores y se debe dar clic en la etiqueta con el icono de “Nuevo contrato” para que se muestren los campos a llenar para el nuevo contrato.

R10: Editar contrato con proveedores

Descripción:

Para editar un nuevo contrato con Proveedores se debe acceder a nivel organizacional al Módulo Contratación. Luego se debe seleccionar la pestaña Contratos con Proveedores, mostrándose la tabla de los contratos creados, en la misma se debe seleccionar la etiqueta con el icono del “editar” en el contrato que se desee editar para que se muestren los campos a llenar.

R11: Eliminar contrato con proveedores

Descripción:

Para eliminar un nuevo contrato con Proveedores se debe acceder a nivel organizacional al Módulo Contratación. Luego se debe seleccionar la pestaña Contratos con Proveedores, mostrándose la tabla de los contratos creados. En la misma se debe seleccionar la etiqueta con el icono del “eliminar “en el contrato que se desee eliminar.

R12: Adicionar una reclamación

Descripción:

Para adicionar una nueva reclamación en Reclamaciones se debe acceder a nivel organizacional al Módulo Contratación. Luego se debe seleccionar la pestaña Reclamaciones, mostrándose la tabla de las reclamaciones creadas. En la misma se debe seleccionar la etiqueta con el icono de “nueva reclamación” para que se muestren los campos a llenar para nueva reclamación.

R13: Editar una reclamación

Descripción:

Para editar una reclamación en Reclamaciones se debe acceder a nivel organizacional al Módulo Contratación. Luego se debe seleccionar la pestaña Reclamaciones, mostrándose automáticamente una tabla de las reclamaciones creadas. En la misma se debe seleccionar la etiqueta con el icono del “editar” en la reclamación que se desee editar, para que se muestren los campos a llenar.

R14: Eliminar una reclamación

Descripción:

Para eliminar una reclamación en Reclamaciones se debe acceder a nivel organizacional al Módulo Contratación. Luego se debe seleccionar la pestaña Reclamaciones, mostrándose automáticamente la tabla de las reclamaciones creadas. En la misma se debe seleccionar la etiqueta con el icono del “eliminar” en la reclamación que se desee eliminar.

R15: Adicionar estado de reclamación

Descripción:

Para adicionar un nuevo estado reclamación en Estados de Reclamaciones se debe acceder a nivel organizacional al Módulo Configuración. Luego se debe seleccionar la pestaña Estados de Reclamaciones, mostrándose automáticamente la tabla de los estados de reclamaciones creados. En la misma se debe seleccionar la etiqueta con el icono de “nuevo estado de reclamaciones” para que se muestren los campos a llenar para crear un nuevo estado.

R16: Editar estado de reclamación

Descripción:

Para editar un estado de reclamación en Estados de Reclamaciones se debe acceder a nivel organizacional al Módulo Configuración. Luego se debe seleccionar la pestaña Estados de Reclamaciones, mostrándose automáticamente la tabla de los estados de reclamaciones creados. En la misma se debe seleccionar la etiqueta con el icono de “editar” en el estado de reclamación que se desee editar, para que se muestren los campos a llenar.

R17: Eliminar estado de reclamación

Descripción:

Para eliminar un estado reclamación en Estados de Reclamaciones se debe acceder a nivel organizacional al Módulo Configuración. Luego se debe seleccionar la pestaña Estados de Reclamaciones, mostrándose automáticamente la tabla de los estados de reclamaciones creados. En la misma se debe seleccionar la etiqueta con el icono de “eliminar” en el estado de reclamación que se desee eliminar.

R18: Exportar cronograma del proyecto

Descripción:

El usuario necesita exportar el cronograma del proyecto, para ello debe acceder a la ruta: Planificación/Cronograma (a nivel de proyecto).

El sistema debe brindar la opción de “Exportar a” que permite que un usuario pueda exportar el cronograma del proyecto a los formatos CSV, PDF.

R19: Adicionar tarea en el cronograma del proyecto

Descripción:

El sistema debe permitir al usuario adicionar tareas dentro del cronograma, para ello debe acceder a la ruta: Planificación/Cronograma (a nivel de proyecto).

R20: Adicionar subtarea en el cronograma del proyecto

Descripción:

El sistema debe permitir al usuario adicionar una subtarea dentro del cronograma, para ello debe acceder a la ruta: Planificación/Cronograma (a nivel de proyecto).

R21: Modificar tarea en el cronograma del proyecto

Descripción:

El sistema debe permitir al usuario gestionar el formulario de cada tarea. Para acceder a los formularios se selecciona la tarea y a la derecha del cronograma se visualiza el mismo.

R22: Importar campos personalizados

Descripción:

Para Importar campos personalizados se debe acceder a nivel organizacional al Módulo Configuración. Luego se debe seleccionar la pestaña Importar campos personalizados y se da clic en la etiqueta Juego de datos para que se desplieguen los campos, se debe seleccionar el dato UCI para que automáticamente se muestre el botón examinar, a través del cual se importa un archivo de tipo CSV que contiene los campos personalizados.

R23: Configurar campos personalizados

Descripción:

Para configurar campos personalizados se debe acceder a nivel organizacional al Módulo Configuración. Luego se debe seleccionar la pestaña Despliegue y se da clic en el botón Configurar campos personalizados y automáticamente se muestra un mensaje color verde en la parte superior que indica que se actualizaron correctamente todos los campos añadidos.

R24: Visualizar diagrama de Gantt

Descripción:

El sistema debe permitir al usuario visualizar las actividades del proyecto a través del diagrama de Gantt, para ello debe acceder a la ruta: Planificación/Peticiones (a nivel de proyecto) y selecciona la opción Gantt.

R25: Visualizar resumen de estado de proyecto

Descripción:

Cuando el usuario acceda a la ruta: Ejecución /Estado del Proyecto (a nivel de proyecto) y seleccionar la pestaña “Resumen de Estado “

R26: Listar las peticiones de calidad de los proyectos sincronizados

Descripción:

El usuario debe permitir listar las peticiones de calidad de todos los proyectos sincronizados, para ello cuando acceda a la ruta: Ejecución/Gerencial (a nivel organizacional), el sistema debe permitir que se visualice la información de las peticiones de calidad de los proyectos.

R27: Actualizar los datos de las planificaciones de calidad

Descripción:

Cuando el usuario acceda a la ruta: Ejecución/Gerencial (a nivel organizacional) y seleccionar la opción ¨Actualizar Planificaciones de calidad¨.

R28: Actualizar datos de las no conformidades

Descripción:

El sistema debe permitir actualizar los datos de las no conformidades de calidad de los proyectos sincronizados, para ello debe acceder a la ruta: Ejecución/Gerencial (a nivel organizacional) y seleccionar la opción ¨ Actualizar no conformidades. ¨

R29: Actualizar incidencias de soporte

Descripción:

El sistema debe permitir actualizar los datos de las no conformidades de calidad de los proyectos sincronizados, para ello debe acceder a la ruta: Ejecución/Gerencial (a nivel organizacional) y seleccionar la opción ¨ Actualizar incidencias de soporte. ¨

R30: Generar plan de datos

Descripción:

El sistema debe permitir al usuario generar el plan de datos, para ello debe acceder a la ruta: Gestión Documenta/Plan de Datos (a nivel de proyecto) y seleccionar la opción “Generar plan de datos” con el objetivo de obtener automáticamente el plan de datos de los artefactos de las tareas.

R31: Visualizar reporte Estado del cliente

Descripción:

El sistema debe permitir al usuario visualizar el reporte Estado del cliente, para ello debe acceder a la ruta: Ejecución/Reportes/Gestión de Alcance/Gestión de interesados (a nivel de organización).

R32: Exportar reportes a PDF

Descripción:

El sistema debe permitir al usuario exportar los reportes en formato .pdf, para ello se debe acceder a la ruta: Ejecución/Reportes/Gestión de Alcance (a nivel de proyecto) y seleccionar la opción Exportar a: PDF.

2.5. Relación Requisito-Tareas

La tabla muestra la asignación de las tareas a los requisitos, dependiendo de la relación existente. Se recoge la información de los requisitos, el identificador, el nombre, el tiempo determinado para su implementación y el conjunto de tareas que se relacionan con cada uno de estos.

Identificador de Requisito (Id.), Nombre de Requisito (NR), Clasificación(C), Tiempo estimado (T), Tareas Relacionadas (TR)

Tabla. 4 Relación Requisito-Tareas

Id.	**NR**	**C**	**T**	**TR**
R1	Filtrar planificación de calidad	Funcional	7	T20, T21, T23, T24
R2	Filtrar no conformidades de pruebas	Funcional	7	T20, T21, T22, T23, T24
R3	Listar no conformidades de revisión	Funcional	7	T20, T21, T23, T24
R4	Filtrar no conformidades de revisión	Funcional	7	T20, T21, T23, T24
R5	Adicionar contrato con clientes	Funcional	7	T20, T21, T23, T24
R6	Editar contrato con clientes	Funcional	7	T1, T2, T3, T4, T5
R7	Eliminar contrato con clientes	Funcional	7	T1, T2, T3, T4, T5
R8	Copiar contrato con clientes	Funcional	7	T1, T2, T3, T4, T5
R9	Adicionar contrato con proveedores	Funcional	7	T1, T2, T3, T4, T5
R10	Editar contrato con proveedores	Funcional	7	T1, T2, T3, T4, T5
R11	Eliminar contrato con proveedores	Funcional	7	T1, T2, T3, T4, T5
R12	Adicionar una reclamación	Funcional	7	T8
R13	Editar una reclamación	Funcional	7	T8, T9
R14	Eliminar una reclamación	Funcional	7	T8, T9
R15	Adicionar estado de reclamación	Funcional	7	T8, T9
R16	Editar estado de reclamación	Funcional	7	T8, T9
R17	Eliminar estado de reclamación	Funcional	7	T8, T9
R18	Exportar cronograma del proyecto	Funcional	6	T1, T2, T3, T4, T5
R19	Adicionar tarea en el cronograma del proyecto	Funcional	6	T1, T2, T3, T4, T5

R20	Adicionar subtarea en el cronograma del proyecto	Funcional	6	T1, T2, T3, T4, T5
R21	Modificar tarea en el cronograma del proyecto	Funcional	6	T1, T2, T3, T4, T5
R22	Importar campos personalizados	Funcional	7	T14, T15, T16, T17, T18, T19
R23	Configurar campos personalizados	Funcional	7	T14, T15, T16, T17, T18, T19
R24	Visualizar diagrama de Gantt	Funcional	6	T14, T15, T16, T17, T18, T19
R25	Visualizar resumen de estado de proyecto	Funcional	3	T14, T15, T16, T17, T18, T19
R26	Listar las peticiones de calidad de los proyectos sincronizados	Funcional	3	T20, T21, T22, T23, T24
R27	Actualizar los datos de las planificaciones de calidad	Funcional	7	T20, T21, T23, T24
R28	Actualizar datos de las no conformidades	Funcional	7	T20, T21, T22, T23, T24
R29	Actualizar incidencias de soporte	Funcional	7	T20, T21, T23, T24
R30	Generar plan de datos	Funcional	3	T20, T21, T22, T23, T24
R31	Visualizar reporte Estado del cliente	Funcional	22	T8, T14, T24
R32	Exportar reportes a PDF	Funcional	10	T8, T14, T24

2.6.Grafo de relación Tarea-Tarea

La Figura 2 muestra el grafo de relación entre las tareas, la ponderación se obtiene a partir de las relaciones entre las tareas, las prioridades de las tareas y la cantidad de requisitos con los que se relaciona cada tarea.

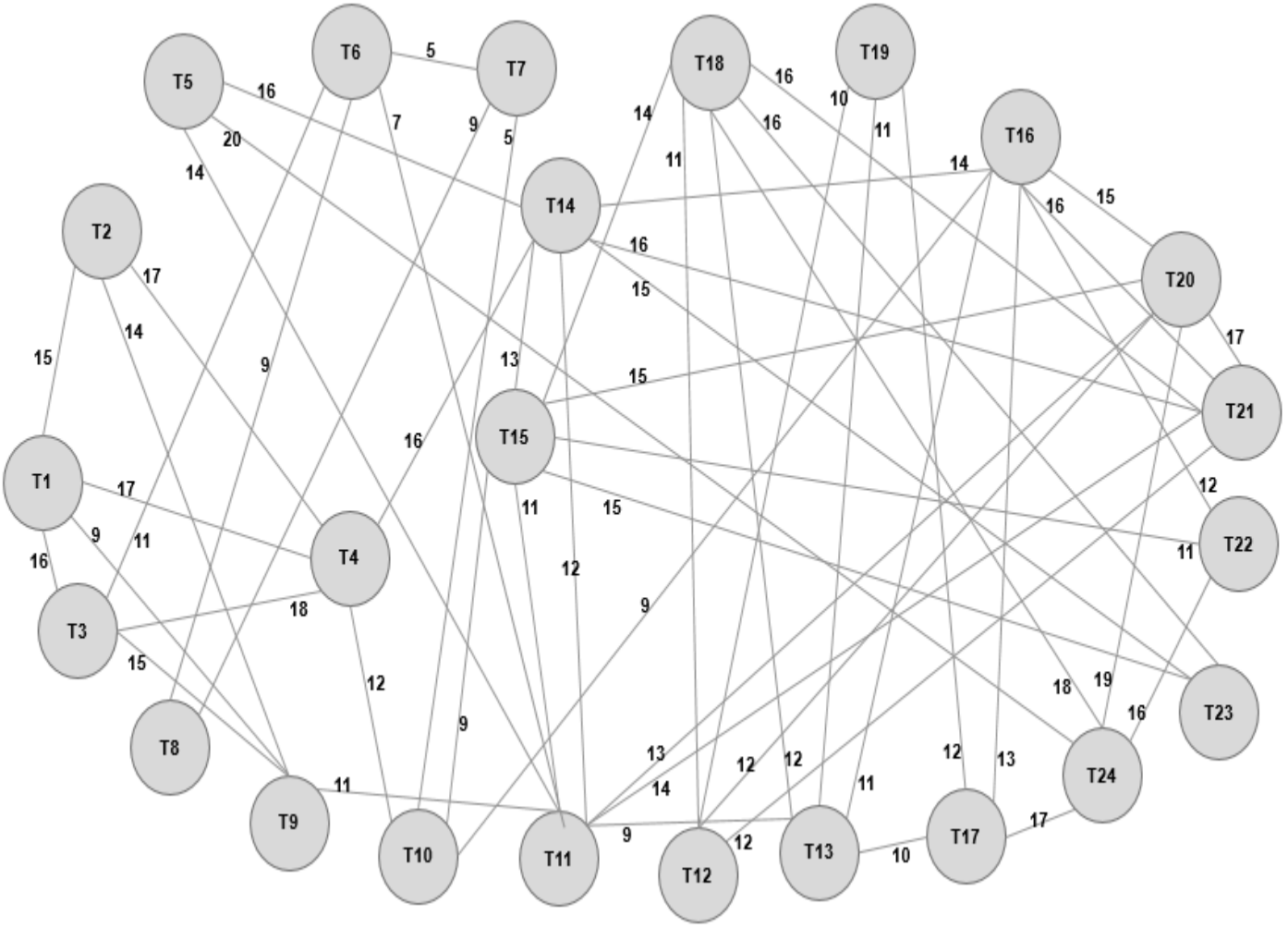

Fig. 2 Grafo ponderado, relación Tarea-Tarea.

2.7.Matriz Booleana

Matriz booleana de las relaciones entre las tareas																								
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24
T1	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T2	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T3	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T4	1	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
T5	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1
T6	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
T7	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T8	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
T9	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
T10	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
T11	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0
T12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
T13	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0
T14	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1	0	1	0
T15	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	1	0	1	0	1	1	0
T16	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	1	1	0	0
T17	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1
T18	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	1	0	1	1
T19	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0
T20	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	1	0	0	1
T21	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1		1	0	1	0	0	0	0
T22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
T23	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0
T24	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0

Fig. 3 Matriz booleana.

2.8.Optimización por colonia de hormigas

Los ACO basan su procedimiento en el comportamiento de las hormigas al trasladarse desde la colonia hasta la fuente de alimento, trazando rutas con rastro de feromona, que es la sustancia que desprenden para por medio de la misma encontrar la ruta más corta, ya que estas son de visión poco desarrollada. La feromona es de fácil evaporación lo que garantiza que por la ruta más corta se haga más recorridos y se desechen las demás. A las hormigas esta sustancia les resulta muy atractiva y entre más fuerte sea la presencia de la sustancia más hormigas transitan en esa dirección.

El ACO genera un gran número de soluciones donde cada hormiga artificial genera una solución en cada iteración lo que sucede hasta llegar a una condición de parada. También se actualiza el rastro de feromona para almacenar la información de las soluciones generadas.

2.8.1 Pasos que sigue el algoritmo

Inicialización

En este paso se inicializan los datos del problema y los parámetros del algoritmo. Se leen los valores que definen el problema y se procesan los datos para obtener información que conlleve a la resolución, la información heurística.

Se define el número de hormigas conociendo que, en dependencia de su tamaño será la exploración del espacio de soluciones y también la memoria que se utilizará y el tiempo de ejecución del mismo.

Se inicializa la información referente a la feromona (T_{ij}).

Inicializar el parámetro de evaporación (ρ).

Inicializar los parámetros de ponderación de influencia de feromonas (α) y de la información heurística (β).

Inicializar las variables que guardan el valor de la información estadística para comprobar el comportamiento del algoritmo según los datos y parámetros utilizados.

Construcción de soluciones

Cada hormiga construye una solución donde cada elección se lleva a cabo por la probabilidad de escoger uno de los posibles nodos dependiendo del rastro de feromonas acumulada y la información heurística asociada a este.

El poder de información heurística permite que la búsqueda se dirija hacia aquellos nodos que más prometen una solución óptima y la feromona inclinan la búsqueda hacia soluciones ya visitadas y de calidad.

Los parámetros α y β permiten ajustar el peso de los factores en el cálculo de la probabilidad.

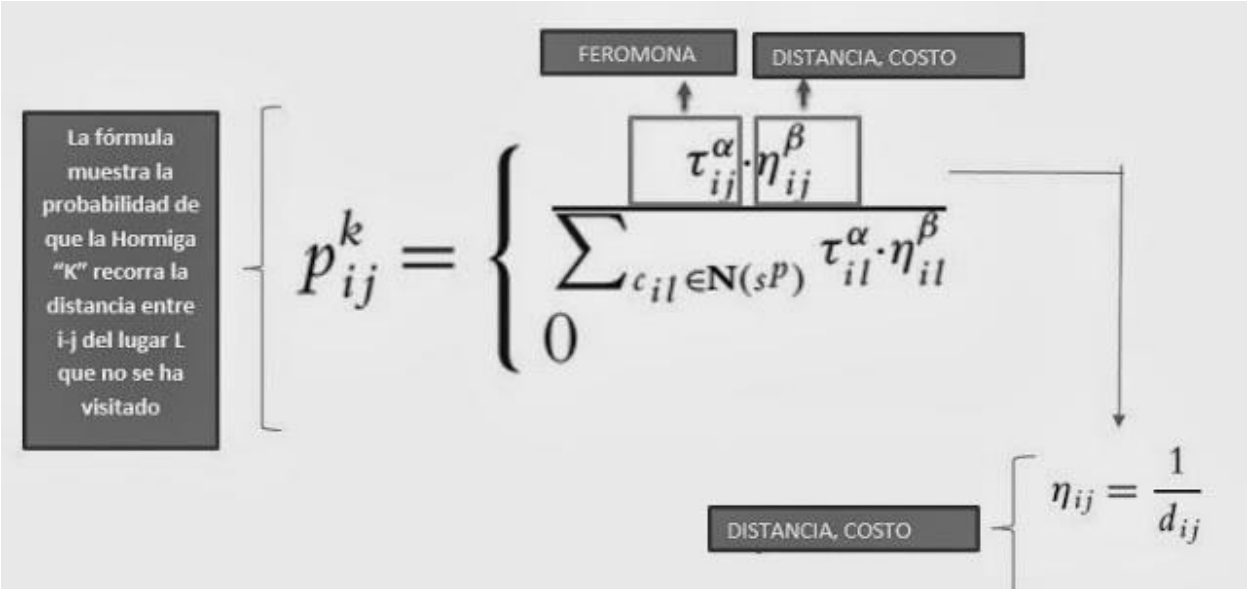

Fig. 4 Probabilidad de escoger un nodo.

Búsqueda local

Esta depende de la calidad de las soluciones iniciales sin que por obligación sean óptimas localmente.

Actualización de feromona

Una vez que cada hormiga construye una solución y finaliza la búsqueda local, el rastro de feromona se actualiza a partir de esas soluciones encontradas. Se deposita la feromona por parte de cada hormiga en cada uno de los elementos del rastro que conforman la solución, aumentando la intensidad del rastro y así la deseabilidad

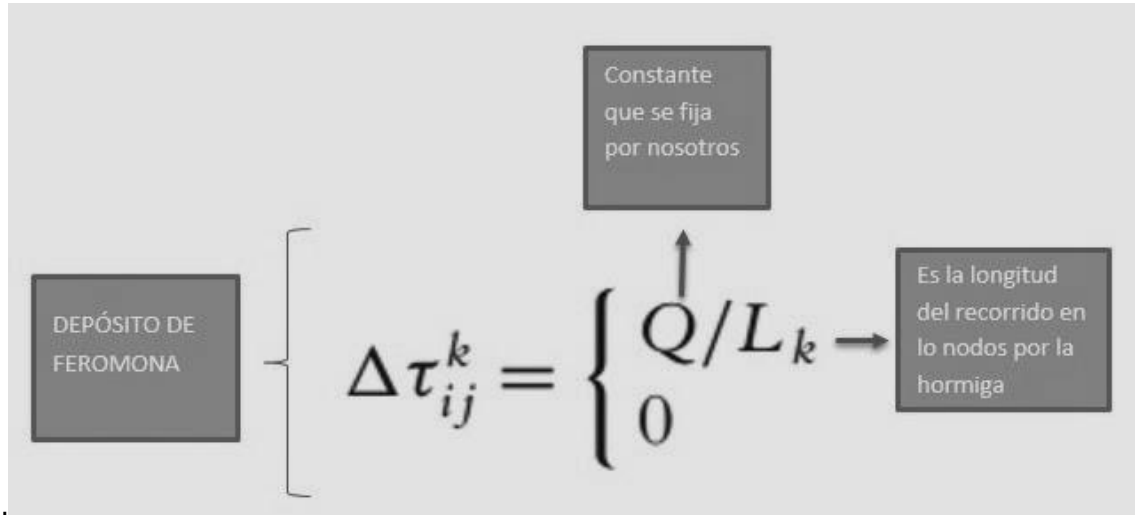

Fig. 5 Depositar feromona

Se realiza la evaporación de feromona en los elementos que componen el rastro, atenuando los depósitos iniciales realizados por soluciones de baja calidad.

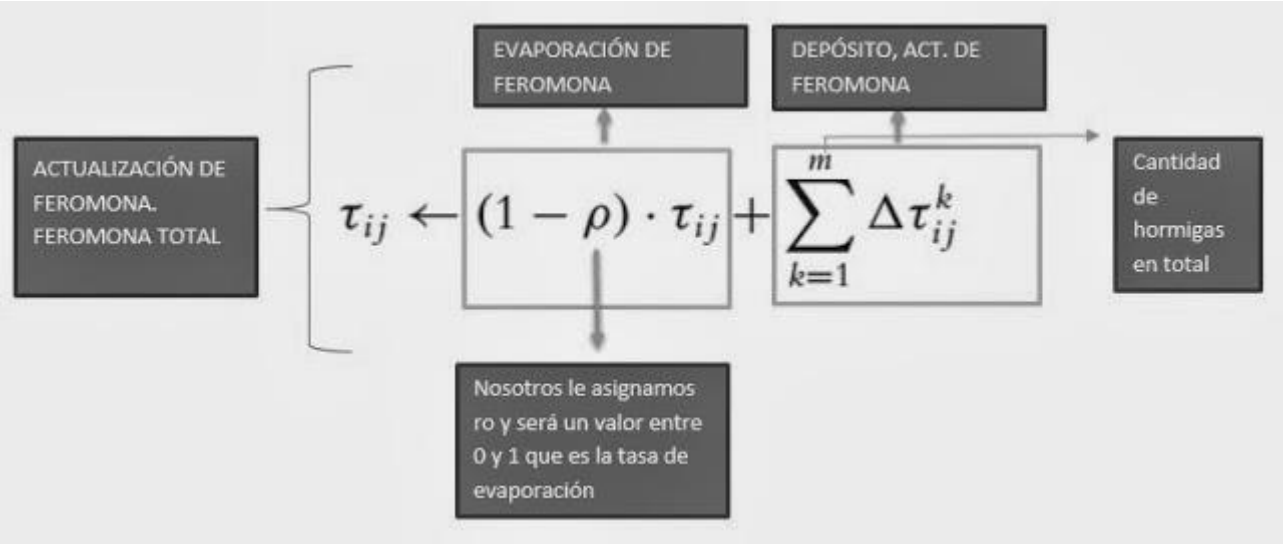

Fig. 6 Actualizar feromona

Criterio de finalización

Puede ser fijando un número de iteraciones o por algún otro criterio de finalización definido.

2.9. Obtención de requisitos

Los requisitos según Sommerville es una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este, es una definición detallada y formal de una funcionalidad del sistema. Estos están expuestos en dos niveles: de usuario y de sistema. Los de usuario son declaraciones en el lenguaje natural y en diagramas de los servicios que el sistema espera proporcione y las restricciones bajo las cuales debe funcionar. Los de sistema por otra parte establecen con detalle las funciones, servicios y restricciones operativas del sistema. Divididos estos por categorías: Requisitos funcionales (RF), Requisitos no funcionales (RNF) y Requisitos de dominio (RD).

Los RF están dados por las funcionalidades básicas de la aplicación como son:

- Adicionar tarea.
- Modificar tarea.
- Eliminar tarea.
- Listar tareas.
- Adicionar requisito.
- Modificar requisito.
- Eliminar requisito.

- Listar requisito.
- Generar secuencia de tareas optimizadas.
- Exportar resultados.

Capítulo 3: Propuesta de Solución

Introducción

La solución propuesta consta de una secuencia de actividades mostradas en el capítulo 2 relacionadas con la relación tarea-requisito. La misma da lugar a un grafo ponderado que constituye la base de aplicación del ACH. Esta solución es validada mediante métodos experimentales.

3.1. Clases definidas

Luego de varios análisis de los datos y la información obtenida, así como del algoritmo de optimización en cuestión se definieron algunas clases como son: controladora, tarea, requisito, hormiga, proyecto.

Clase Controladora:

Es la clase que controla todo el proceso de gestión de las funcionalidades de la aplicación. Esta consta de un proyecto, el cual contiene la información de los requisitos y las tareas sobre las cuales trabaja la aplicación en el proceso de optimización, una lista de hormigas que se encargaran de la ejecución del algoritmo, una lista de recorridos donde se almacenan las tareas recorridas en cada ejecución y un grafo con la representación de las relaciones entre las tareas.

Clase Proyecto:

El proyecto se construye a partir de atributos como son el identificador del proyecto, su nombre, la lista de requisitos y la lista de tareas del proyecto.

Clase Tarea:

La tarea está compuesta por un identificador, una clasificación según del tipo de tarea, una prioridad con respecto a otras tareas y el texto que define la actividad.

Clase Requisito:

Un requisito tiene un identificador, consta de un texto que define el requisito, una clasificación y el tiempo que demora implementarlo.

Clase Hormiga:

Es la clase que representa el agente artificial, incluye una lista de tareas visitadas y otra de no visitadas, además de un identificador que corresponde con la tarea en la que comienza su recorrido en la ejecución del algoritmo.

Además de estas clases declaradas se utilizan otras clases abstractas para la gestión de los datos necesarios en la optimización, ejemplo de estas son, Lista, Grafo, y para implementar el grafo se declaran otras dos clases llamadas vértice y arista.

Cuando el usuario necesite generar y optimizar la secuencia de tareas que son asignadas a los requisitos para que estos sean implementados con calidad y en correspondencia con el tiempo estimado para tal, el usuario debe respetar estrictamente la estructura definida en el archivo base_conocimiento.txt que la aplicación utiliza para gestionar la información que procesa. En este archivo debe introducir los datos de las tareas y los requisitos a utilizar, así como las relaciones entre las tareas para formar el grafo y también las relaciones entre las tareas y los requisitos. De igual manera la aplicación brinda la posibilidad de adicionar, modificar o eliminar un requisito, una tarea y las relaciones correspondientes.

3.2. Proceso de optimización

El proceso comienza cuando el usuario inicia la aplicación, ha realizado los cambios en los datos cargados del archivo que contiene la información de las tareas, los requisitos y las relaciones o decide no realizar cambio alguno, se dirige a la pestaña de proyecto, crea el proyecto con un nombre y los datos anteriormente cargados. A partir de allí se crea el grafo correspondiente al proyecto sobre el cual se trabaja, las tareas, los requisitos y las relaciones, tarea-tarea y requisito-tarea. Las tareas son los vértices del grafo, las aristas son las relaciones entre las tareas, el peso de las aristas es determinado automáticamente por la media aritmética entre los valores de ponderación de los vértices que la conforman. Luego, da un click en el botón Generar Secuencia de Tareas, se realiza la optimización por medio del ACH y se muestra la lista de tareas generada.

Diagrama de clases

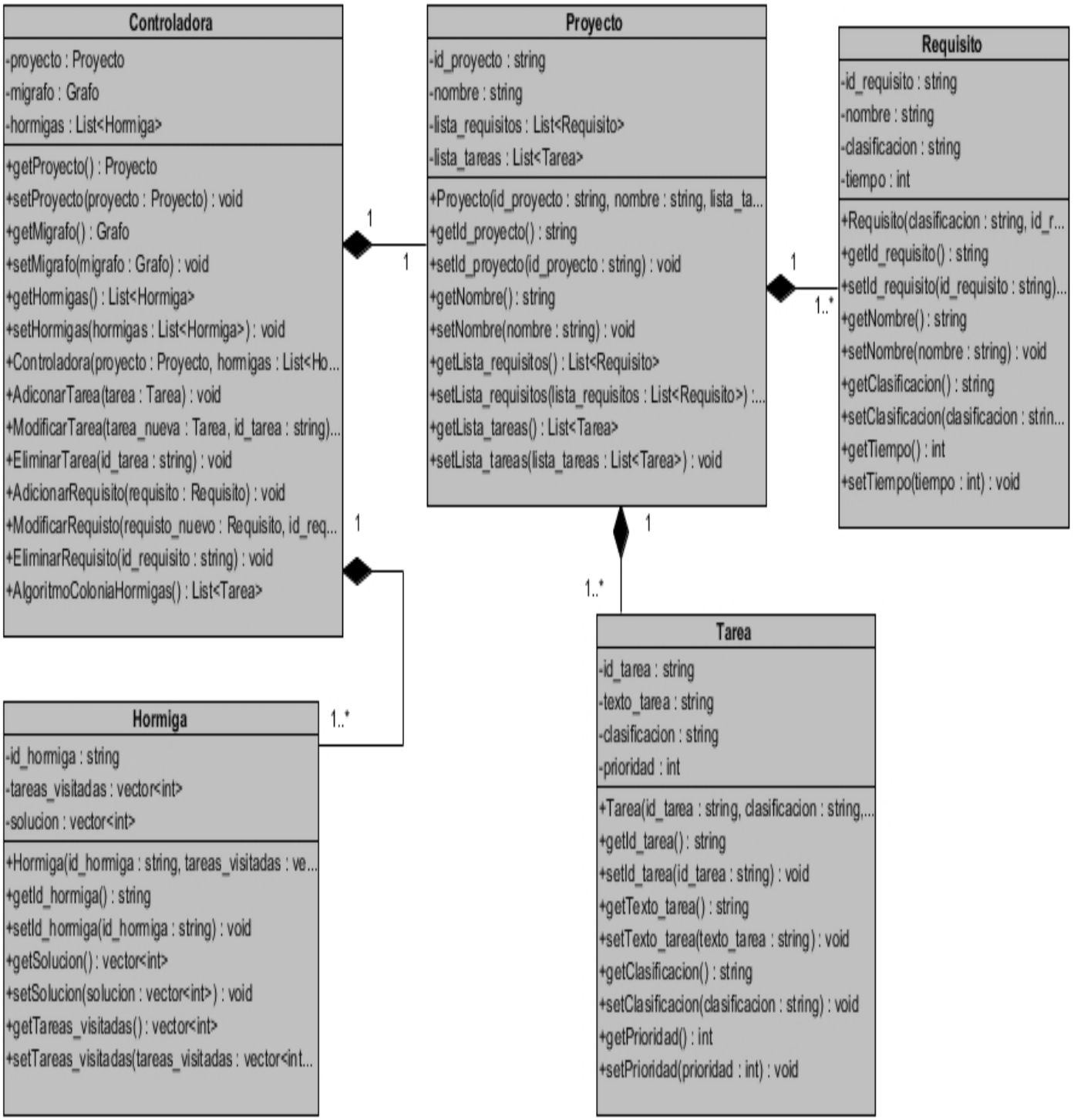

Fig. 7 Diagrama de clases

3.2.1 Funcionalidades de la aplicación

Las funcionalidades de la aplicación están enfocadas al principio de funcionamiento del Algoritmo Colonia de Hormigas, aquí se describen la mayor influencia en cuanto a la optimización, las mismas son:

Cargar Archivo: permite al usuario cargar el archivo que contiene la información a procesar para lograr la optimización de las tareas relacionadas con los requisitos del proyecto en el que se trabaja.

Añadir tarea: el usuario entra los datos de la tarea a añadir, incluye el nombre de la tarea, el identificador o variable que identifica a la tarea, la prioridad y la clasificación, presiona el botón Añadir Tarea y la tarea se añade a la lista de tareas.

Modificar tarea: el usuario elige una tarea de la lista de tareas y presiona el botón Modificar Tarea y los datos de la tarea a modificar pasan automáticamente a la sección de modificación en el área de modificar tarea y se muestra esta ventana.

Eliminar tarea: el usuario selecciona la tarea a eliminar de la lista de tareas. Presiona el botón Eliminar Tarea y se elimina la tarea de la lista.

Añadir requisito: el usuario entra los datos del requisito a añadir, incluye el nombre del requisito, el identificador del requisito, el tiempo necesario para su implementación y su clasificación. Presiona el botón Añadir y el requisito se añade a la lista de requisitos.

Modificar requisito: el usuario elige un requisito de la lista de requisitos y presiona el botón Modificar y los datos del requisito a modificar pasan automáticamente a la sección de modificación en el área de modificar requisito y se muestra esta ventana.

Eliminar requisito: el usuario selecciona el requisito a eliminar de la lista de requisitos. Presiona el botón Eliminar y se elimina requisito de la lista.

Generar secuencia de tareas: el usuario desea generar la secuencia de tareas optimizadas a partir de los requisitos del proyecto y la relación de las tareas con estos. Presiona en el botón Generar Secuencia de Tareas y la aplicación procesa los datos pertenecientes al proyecto y por medio del ACH, procede a la optimización de la secuencia de tareas que se generan y muestra al usuario la más óptima encontrada en dependencia del juego de datos.

La tabla muestra de la obtención de la ponderación del grafo en las aristas.

Tabla. 5 Ponderación de las aristas.

Arista	**Peso de T_1**	**Peso de T_2**	**Peso de Arista**

T_1	T_2	Pr	Ve	RR	$Suma_1$	Pr	Ve	RR	$Suma_2$	$(Suma_1 + Suma_2)/2$
T1	T 4	1	4	10	15	4	5	10	19	17
T6	T11	1	4	0	5	2	8	0	10	7
T16	T14	3	7	4	14	1	7	6	14	14
T5	T24	5	3	10	18	5	5	12	22	20

En la tabla se muestra el procedimiento de cómo se obtiene las ponderaciones de las aristas de forma manual. Proceso que realiza la aplicación de manera automática, para cada una de las aristas, primero se busca la ponderación de cada vértice y luego se haya la media entre cada par de vértices que compone las aristas, cada vértice en el grafo corresponde a una tarea.

En la tabla por temas de organización se trabaja a T_1 *y* T_2 en función del vértice1 y del vértice2 de la arista, respectivamente, *Pr* como la prioridad de dicho vértice, *Ve* como la cantidad de vértices con lo que se relaciona este, *RR* es la cantidad de requisitos con los que se relaciona la tarea en cuestión y $Suma_i$, para $i=1,2$; es la suma de dichos valores *(Pr, Ve, RR).*

3.3. Descripción de los pasos que sigue el algoritmo para optimizar

Una vez iniciado el algoritmo, este inicializa los parámetros de ponderación α y β con valores mayores que cero, de manera tal que se tengan en cuenta la feromona y la visibilidad en la decisión hacia que nodo desplazarse en el siguiente movimiento. Inicializa la matriz de feromonas con un valor de 0.1 en cada arista, la lista de las hormigas para un total de hormigas igual a la cantidad de tareas. Todas las hormigas con el vector de solución vacío y el vector de los vértices visitados con todas las posiciones en cero.

Luego, cada una de las hormigas construye una posible solución iniciando su recorrido en el vértice que se corresponde con su posición en la lista, recorriendo todos los nodos o la mayor cantidad de nodos posibles y añadiendo estos a su vector de solución, siempre y cuando ese nodo no haya sido visitado por ella. Información que está actualizando constantemente en el vector de vértices visitados, el primer nodo en la solución es el nodo de inicio y es además el primer nodo visitado. El nodo a visitar por la hormiga se corresponde con la probabilidad que ofrece el acceso a este desde el nodo en el que se encuentra anteriormente.

La probabilidad de visitar un nodo está dada por la expresión dada en la Figura 4**,** calculada la probabilidad entre el nodo actual y los vecinos, se escoge la mayor probabilidad y se actualiza el nodo actual con el nodo de mayor probabilidad. Se añade este a la solución y se actualiza como visitado, se repite este proceso tanto como sea posible.

El depósito de feromonas ocurre luego de haber encontrado una posible solución. Se deposita una porción de feromona en el camino encontrado como solución (ver expresión mostrada en la Figura 5). La actualización de feromona ocurre luego de este paso para disminuir la influencia de las feromonas en las posteriores soluciones, dejando los caminos menos usados a partir de este momento menos deseables para el algoritmo.

3.4. Interfaces del sistema

La aplicación consta de una ventana en la cual se encuentran las diferentes funcionalidades distribuidas por pestañas correspondientes a Tarea, Requisito, Modificar y Proyecto. En la ventana se encuentra además un menú con la opción de Cargar Archivo que contiene los diferentes datos de entrada de la aplicación. En la pestaña Tarea están las funcionalidades de Añadir Tarea, Modificar Tarea, Eliminar Tarea y se muestra la lista de las tareas cargadas del archivo. En la pestaña Requisito se encuentran las funcionalidades de Añadir, Modificar, Eliminar y además se muestra la lista de requisitos cargados desde el archivo.

El funcionamiento se explicar a través de las interfaces que la conforman.

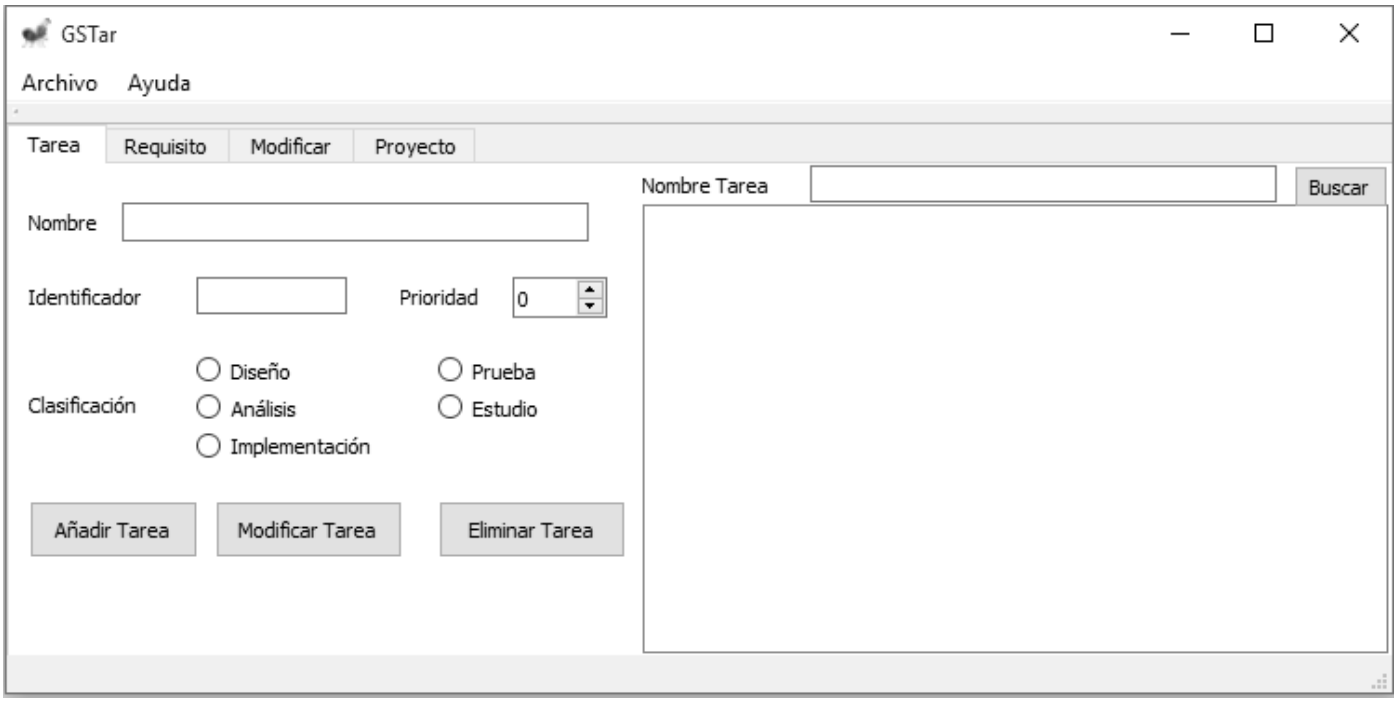

Fig. 8 Interfaz de inicio

La interfaz de inicio, es la primera interfaz que muestra la aplicación una vez iniciada, aquí se puede añadir una tarea. Además, se muestran las pestañas Requisito y Modificar.

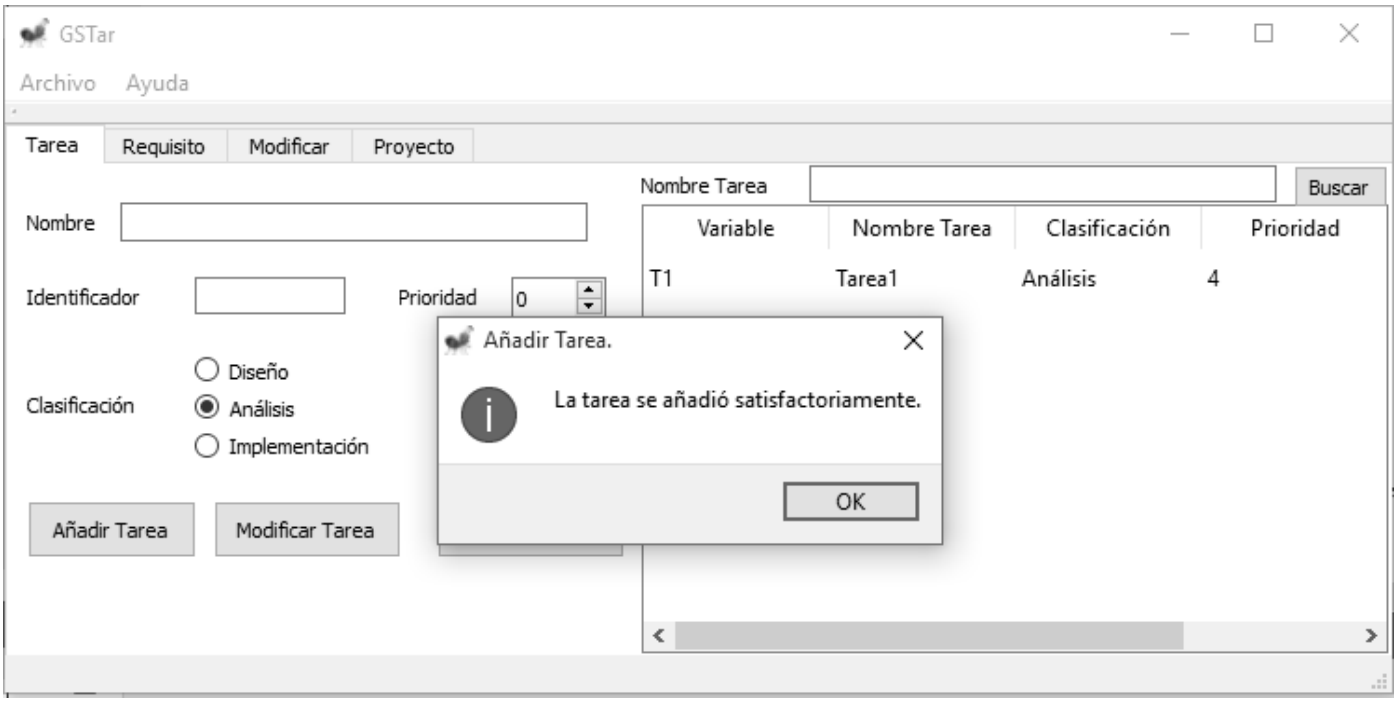

Fig. 9 Interfaz Añadir Tarea

Se observa un ejemplo de la acción de añadir una tarea, el usuario entra datos válidos y da click en el botón Añadir Tarea. Automáticamente la aplicación comprueba la información y si es correcta crea la tarea, la añade a la lista y la muestra en la tabla organizada por sus atributos.

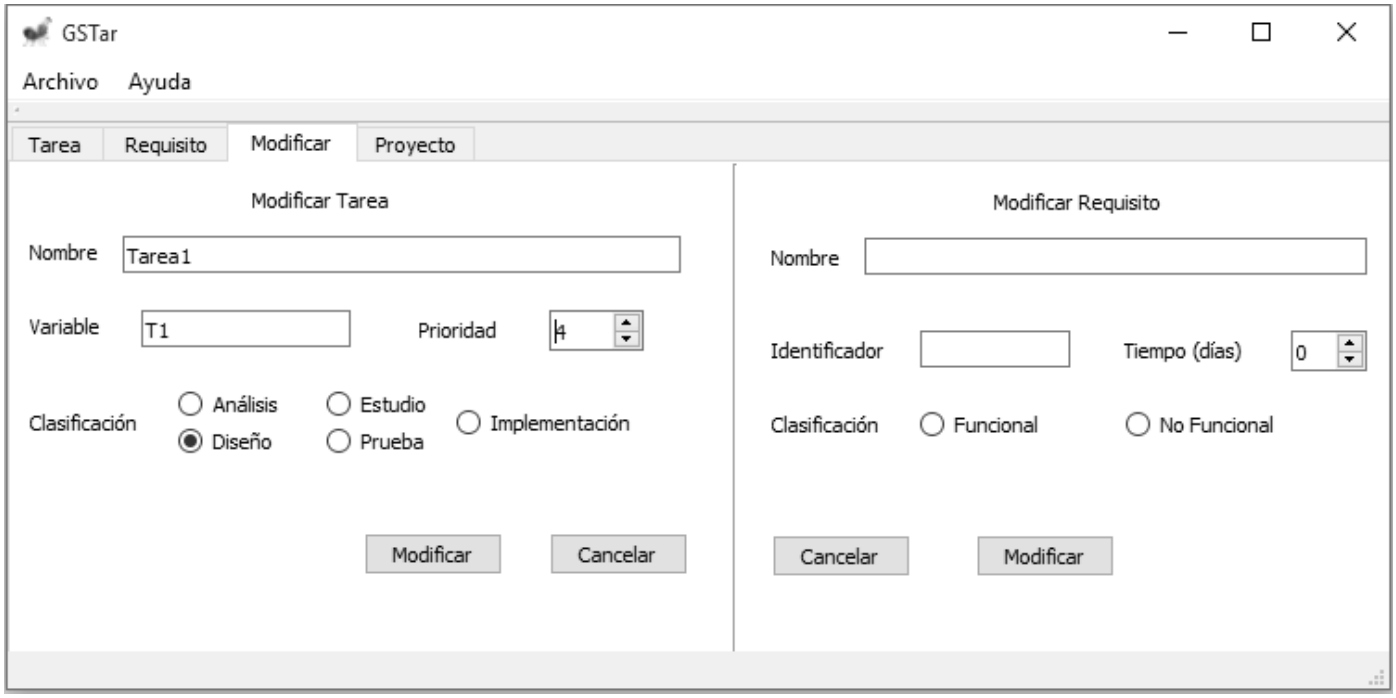

Fig. 10 Interfaz modificar tarea (a).

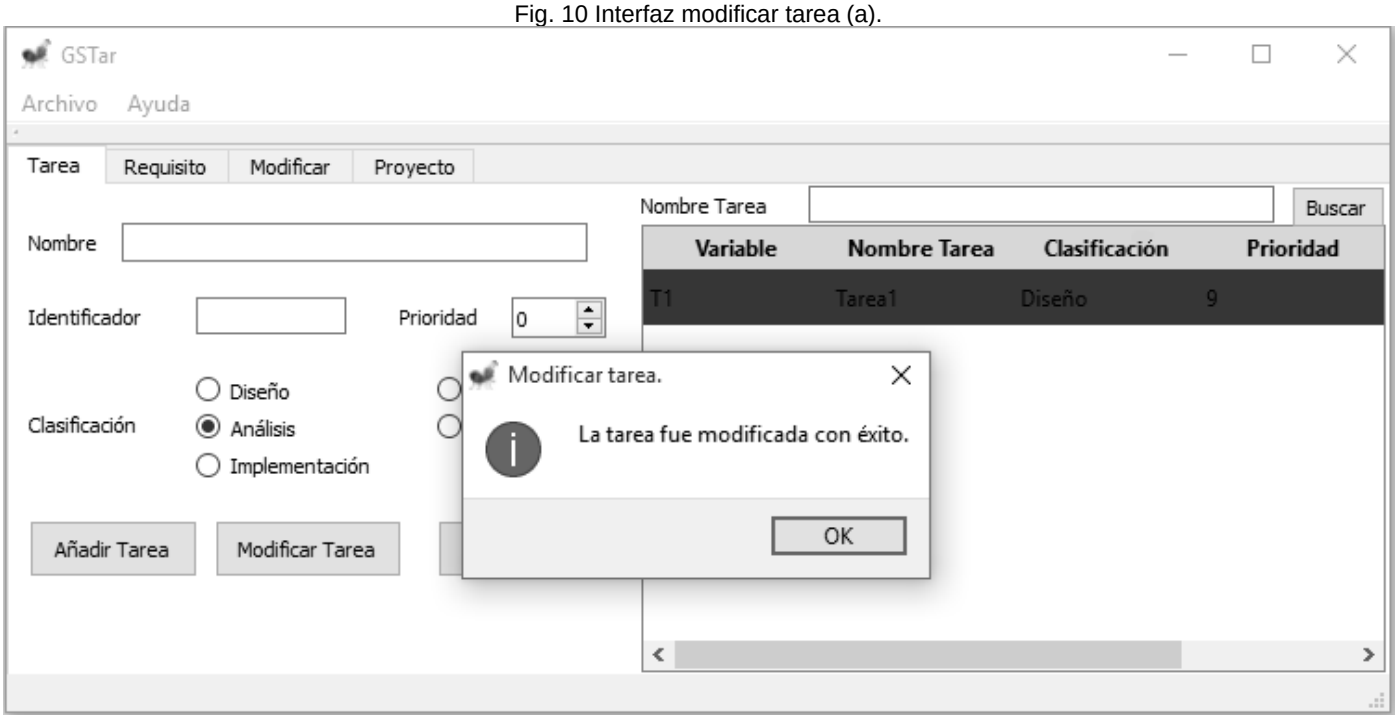

Fig. 11 Interfaz modificar tarea (b).

Se muestra el proceso de modificación de una tarea, el usuario selecciona la tarea en la lista de tareas, da click en Modificar Tarea, los datos de la tarea pasan a la interfaz de modificar en los campos correspondientes, donde se muestran opciones de modificar o cancelar la modificación, si la tarea es modificada, es mostrada en la lista de tareas y se muestra un mensaje de confirmación de que la tarea fue modificada con éxito.

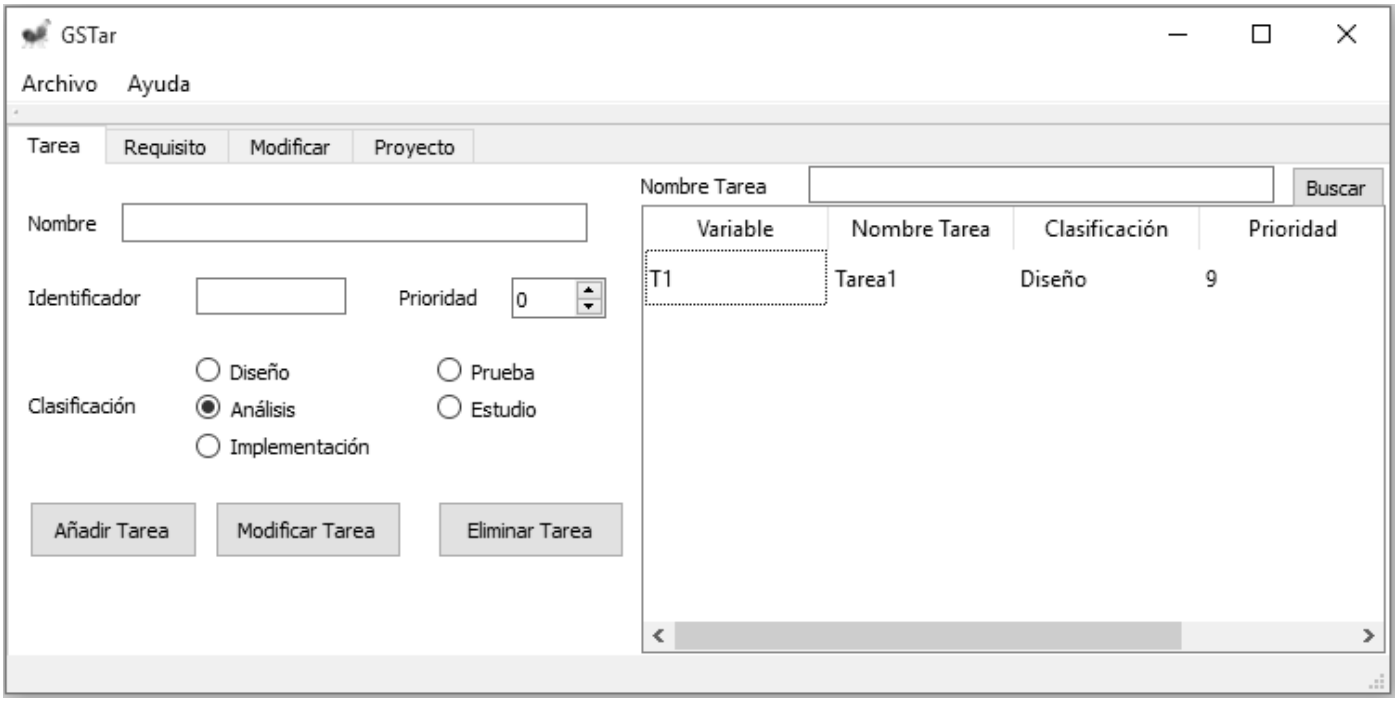

Fig. 12 Interfaz eliminar tarea (a).

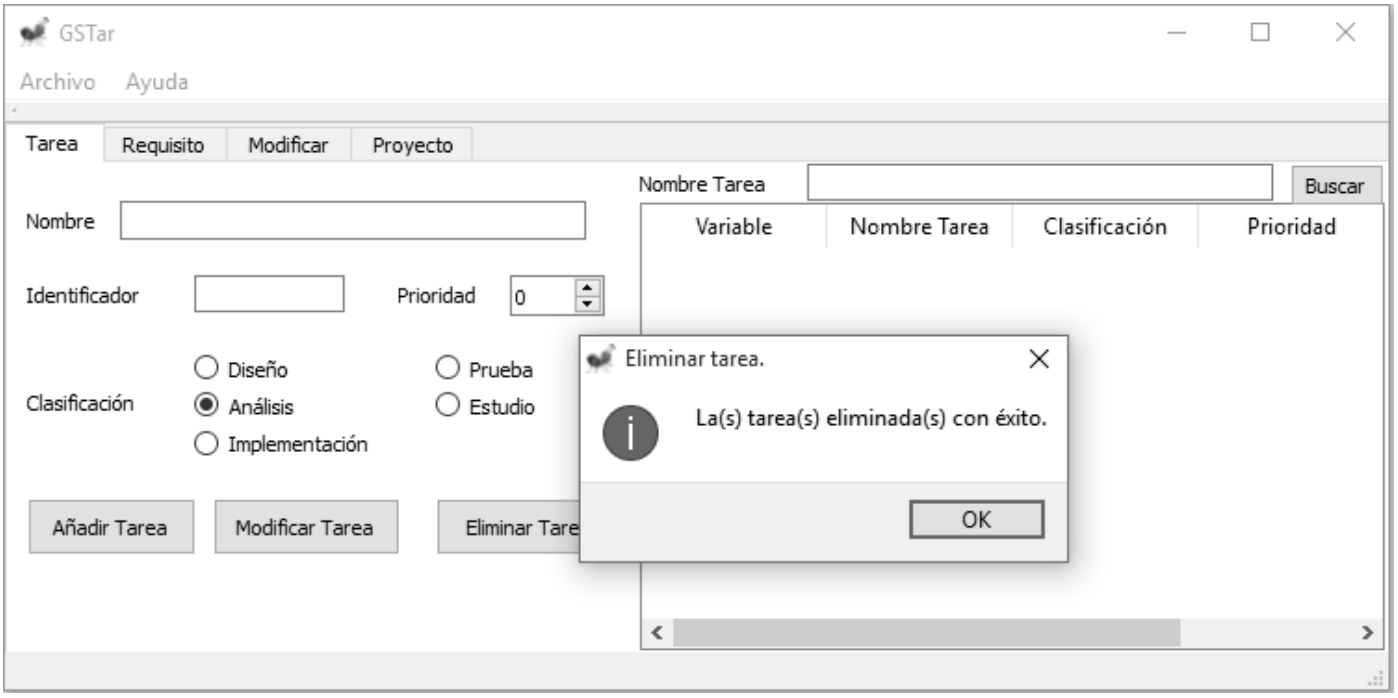

Fig. 13 Interfaz eliminar tarea (b).

La interfaz muestra el proceso de eliminación de una tarea, donde el usuario selecciona la tarea que desea eliminar y da click en el botón Eliminar Tarea, la tarea es eliminada de la lista y se muestra en mensaje de confirmación de que la tarea fue eliminada con éxito.

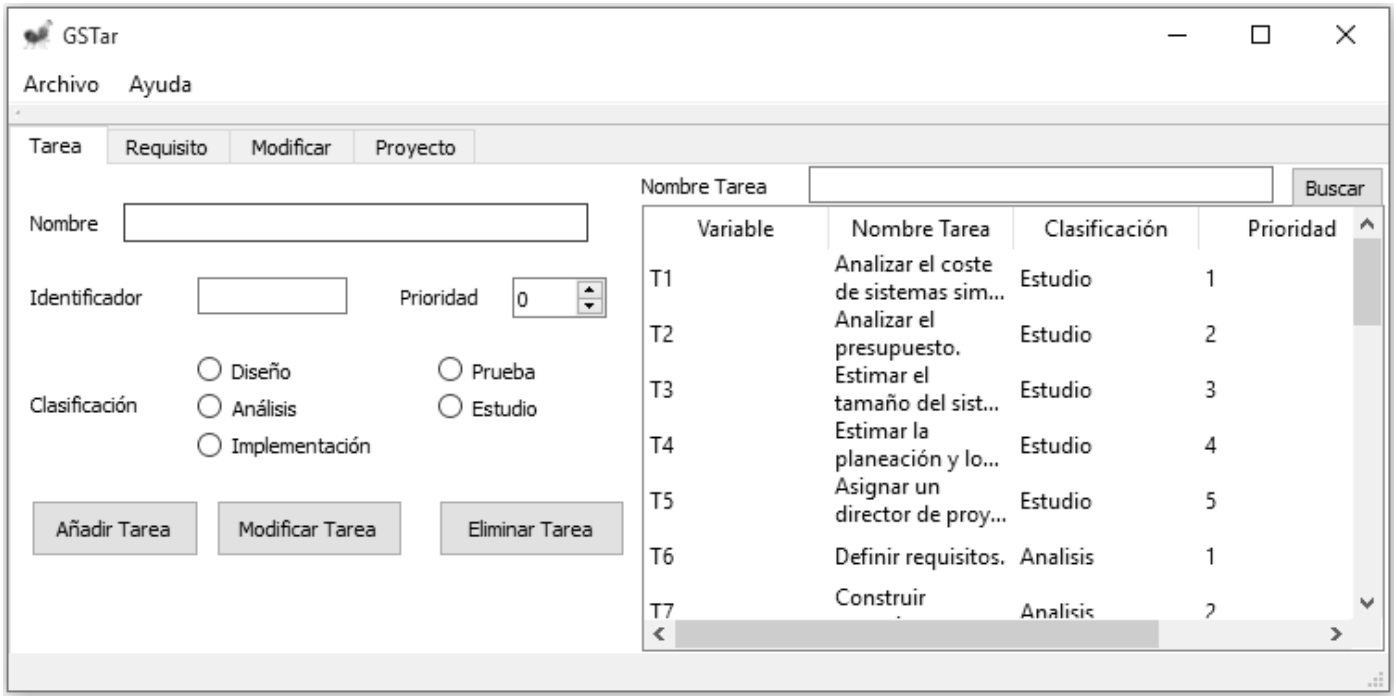

Fig. 14 Interfaz lista de tareas.

La interfaz muestra la lista de tareas cargadas del archivo base.

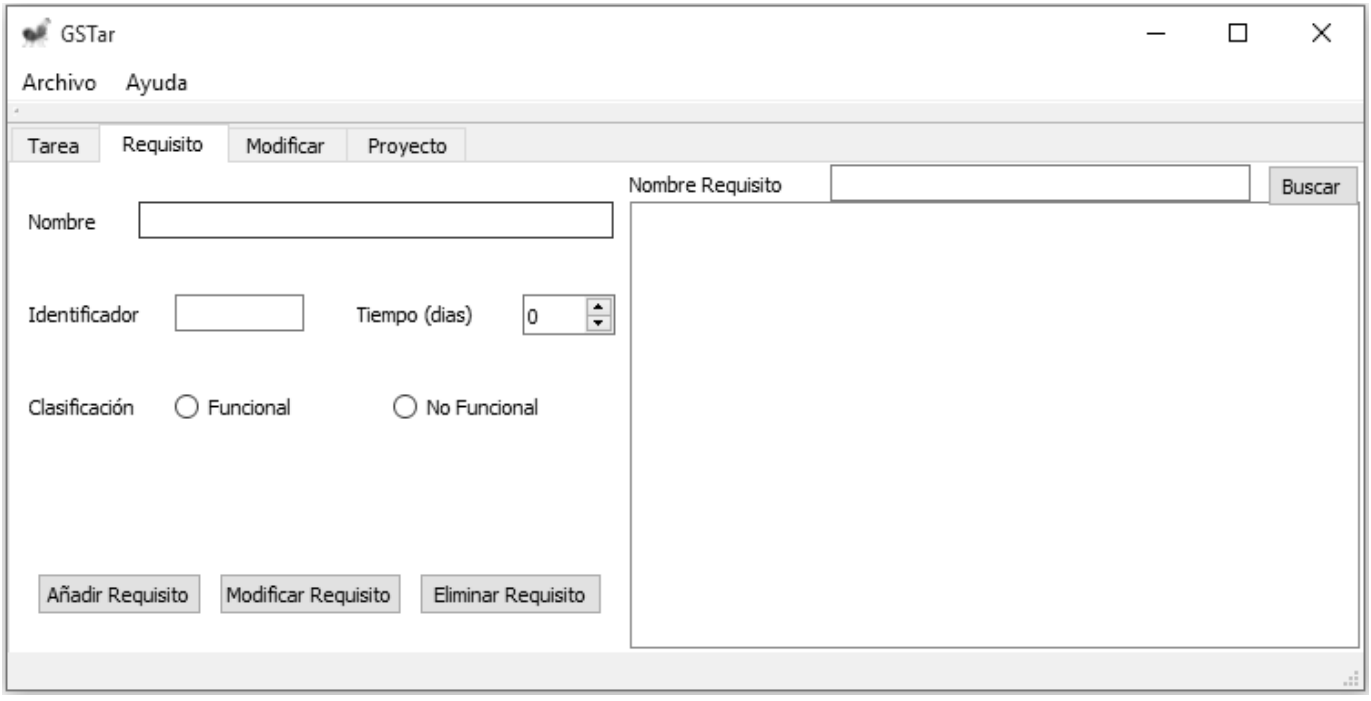

Fig. 15 Interfaz de Requisito

En la interfaz se observan los campos correspondientes para introducir los datos de un nuevo requisito y la opción de Añadir Requisito, Modificar Requisito y Eliminar Requisito. En la clasificación de los

requisitos, la clasificación No Funcional cuando es seleccionada se despliega las clasificaciones Software, Hardware y Soporte, correspondiente a la clasificación de requisitos no funcionales.

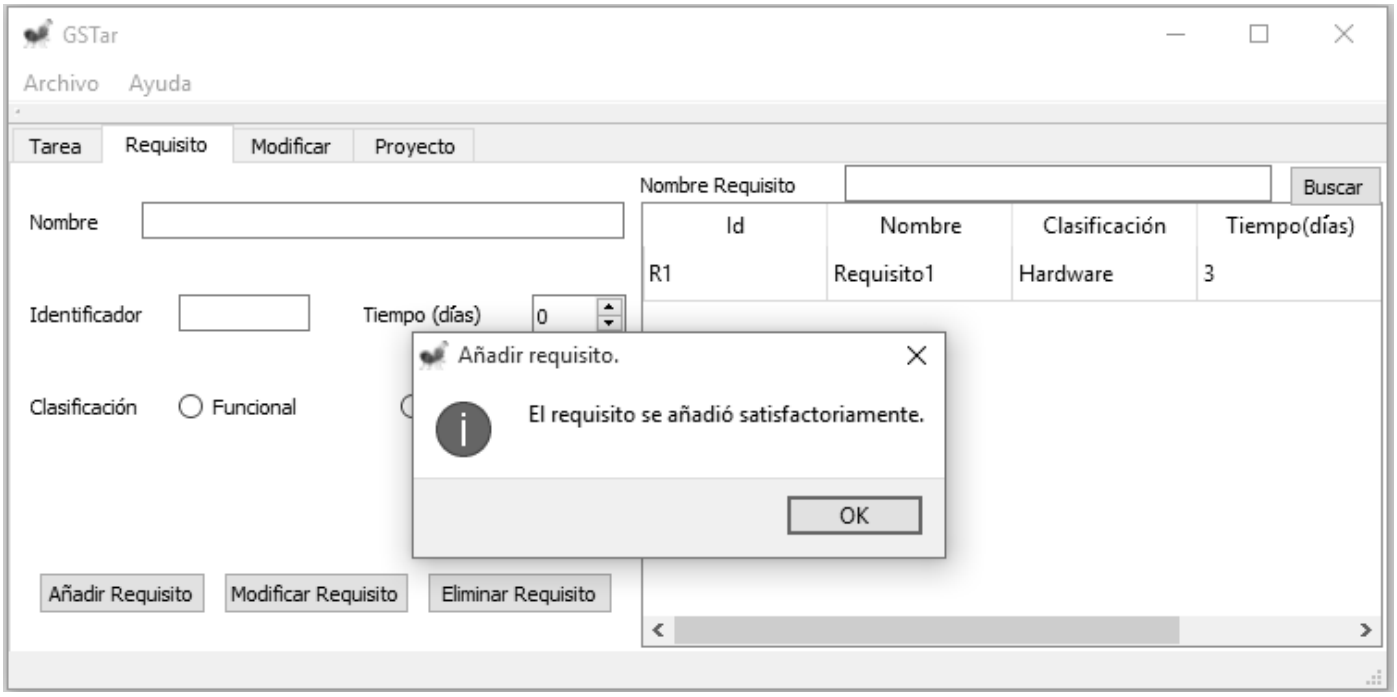

Fig. 16 Interfaz Añadir Requisito

Se observa la acción de añadir un nuevo requisito, el usuario entra datos correctos para un requisito válido y da click en el botón Añadir Requisito. La aplicación comprueba la información, la procesa, añade el requisito a la lista de requisitos y los muestra en la tabla, organizado según sus atributos.

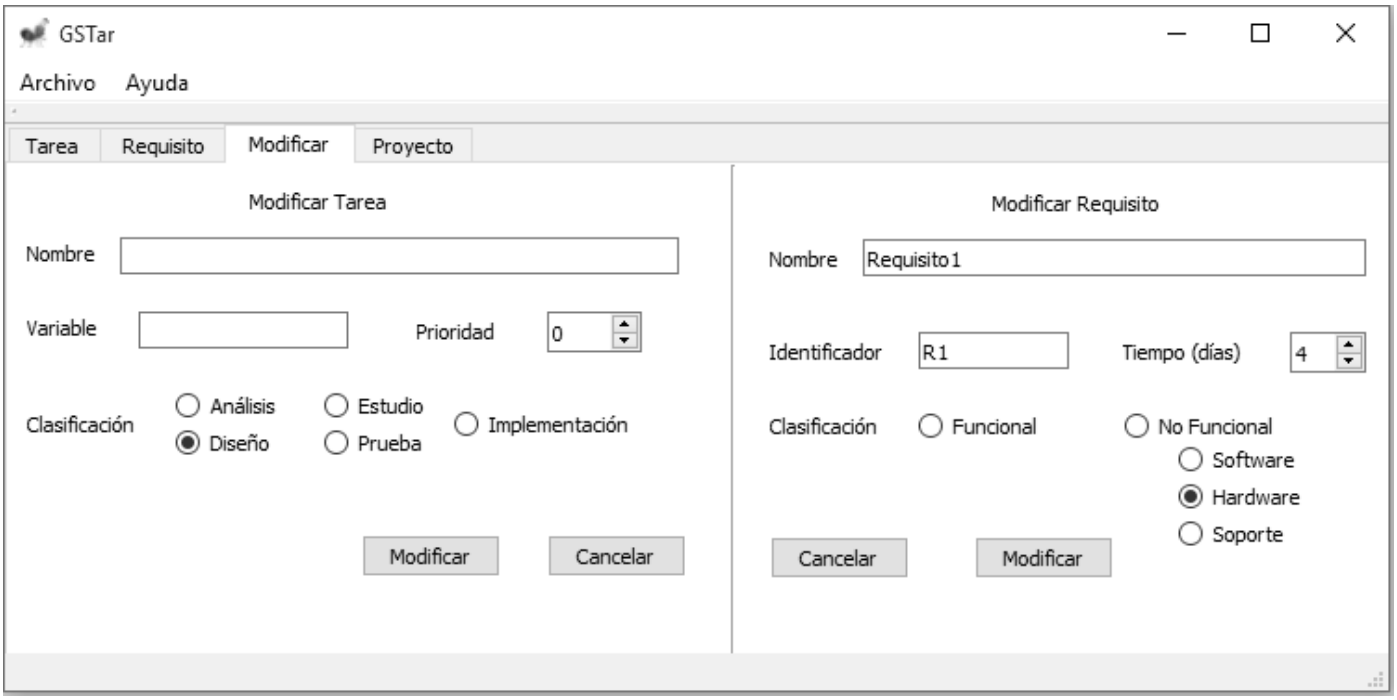

Fig. 17 Interfaz modificar requisito (a).

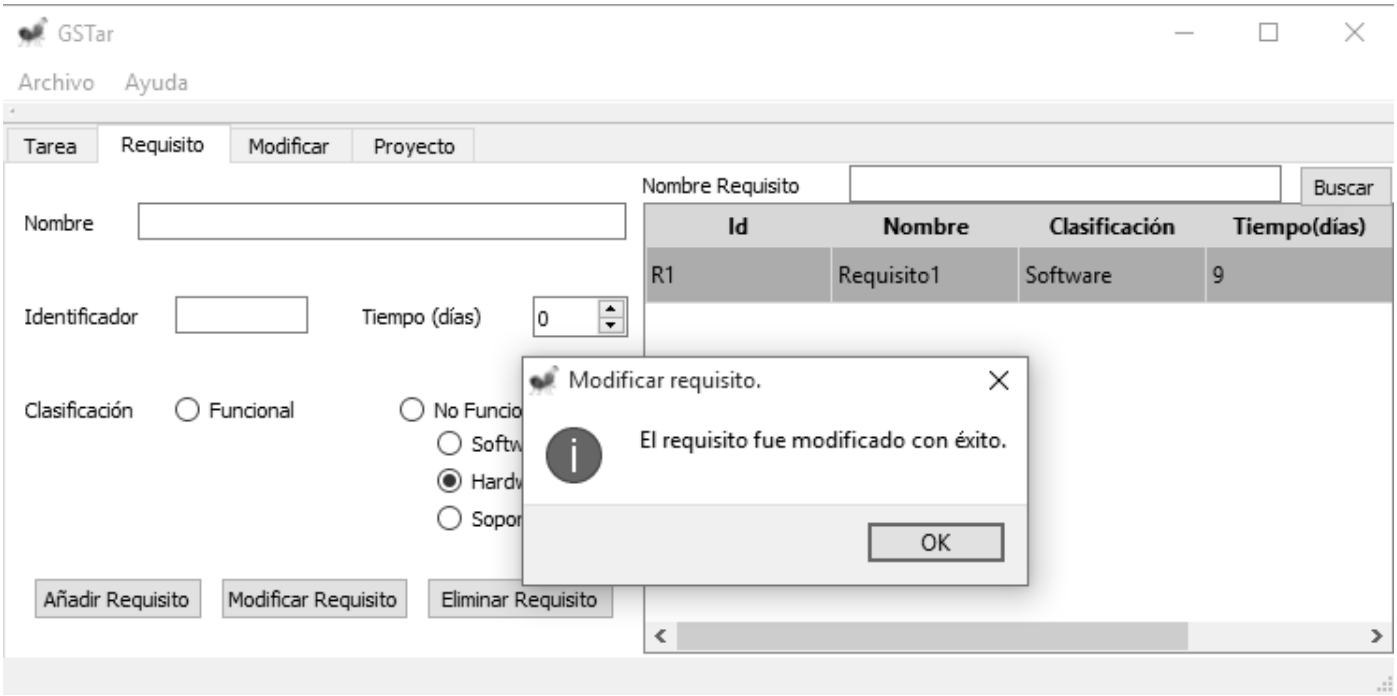

Fig. 18 Interfaz modificar requisito (b).

Se muestra el proceso de modificación de un requisito, el usuario selecciona el requisito en la lista de requisitos, da click en Modificar Requisito, los datos del requisito pasan a la interfaz de modificar en los campos correspondientes, donde se muestran opciones de modificar o cancelar la modificación, si el

requisito es modificado, es mostrado en la lista de tareas y se muestra un mensaje de confirmación de que el requisito fue modificado con éxito.

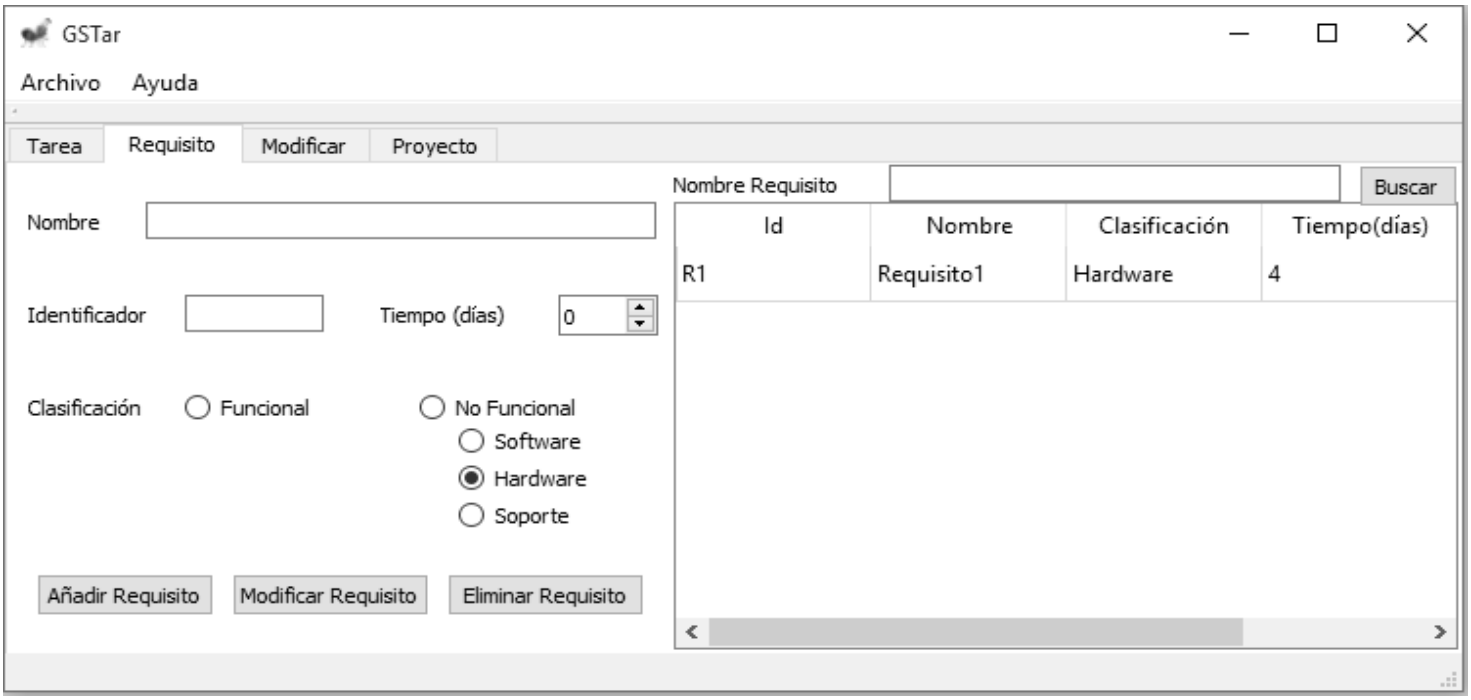

Fig. 19 Interfaz eliminar requisito (a).

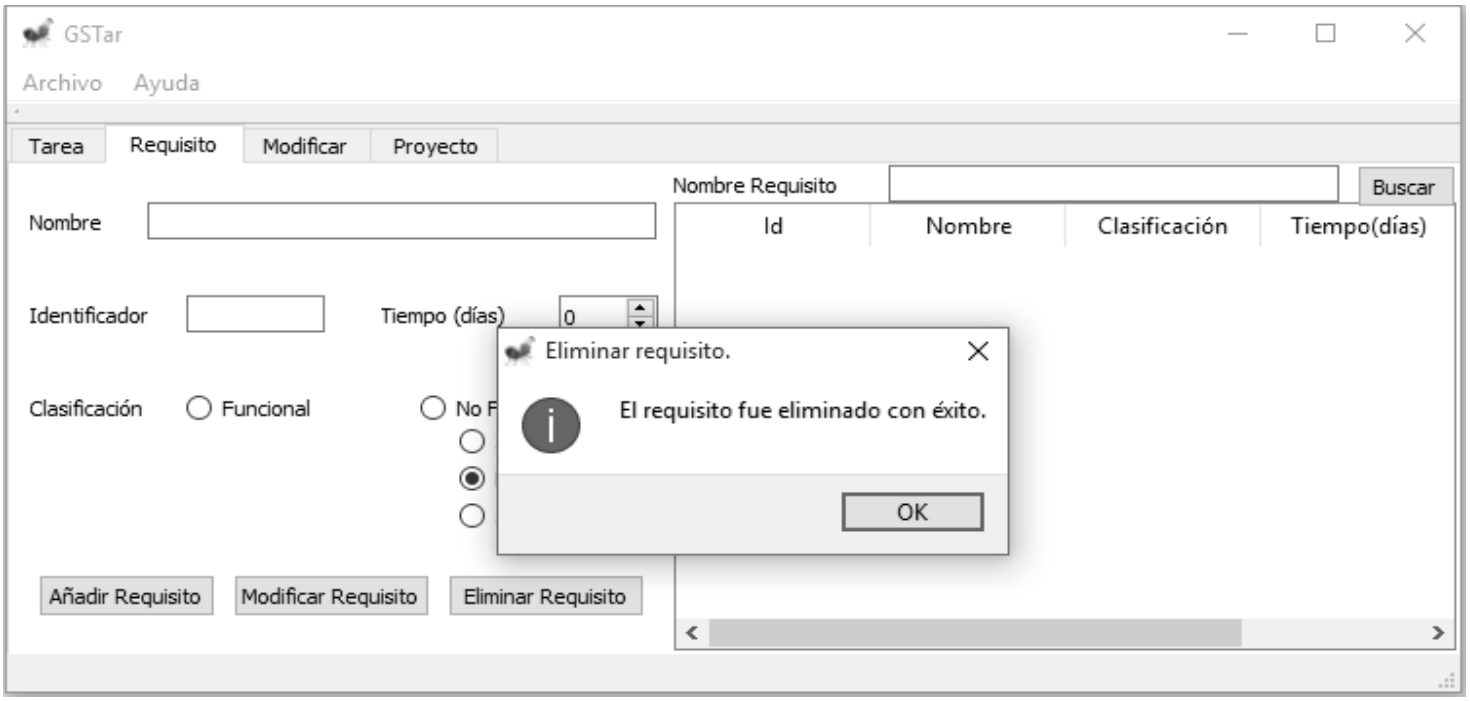

Fig. 20 Interfaz eliminar requisito (b).

La interfaz muestra el proceso de eliminación de un requisito, donde el usuario selecciona el requisito que desea eliminar y da click en el botón Eliminar Requisito, el requisito es eliminado de la lista y se muestra en mensaje de confirmación de que el requisito fue eliminado con éxito.

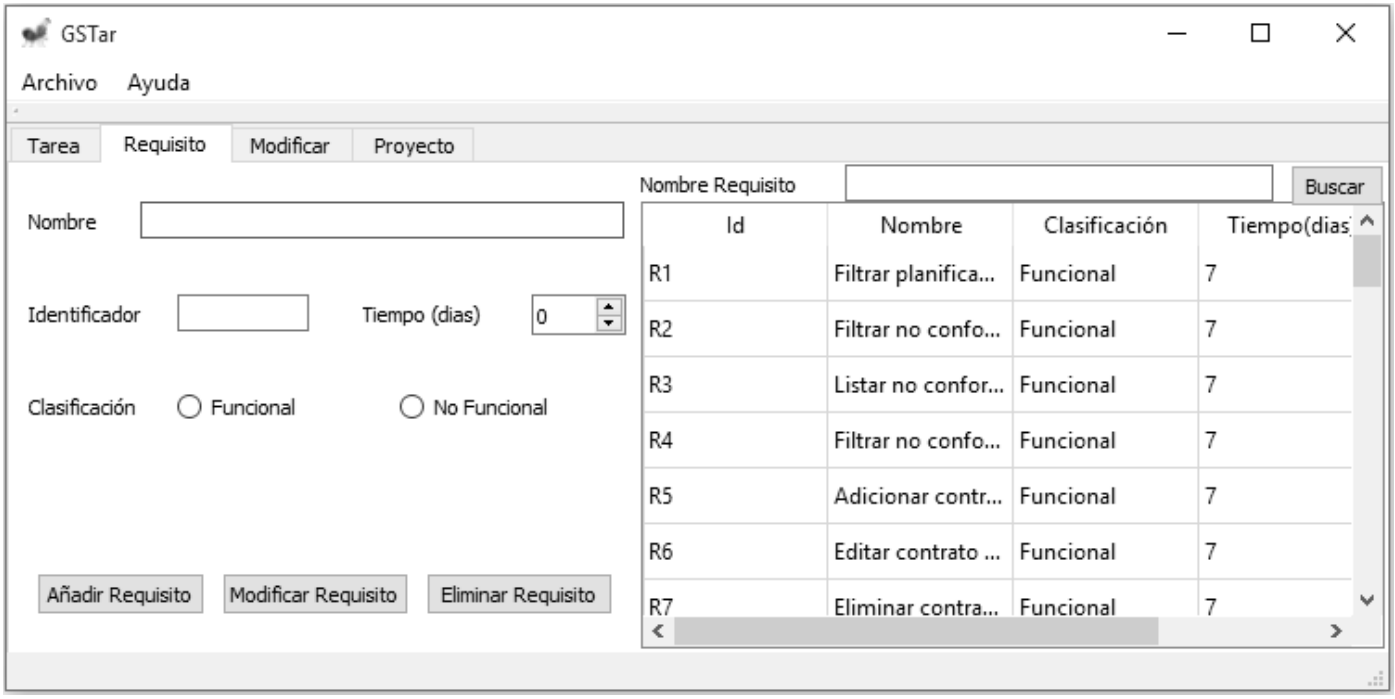

Fig. 21 Interfaz lista de requisitos.

La interfaz muestra la lista de requisitos cargados del archivo base.

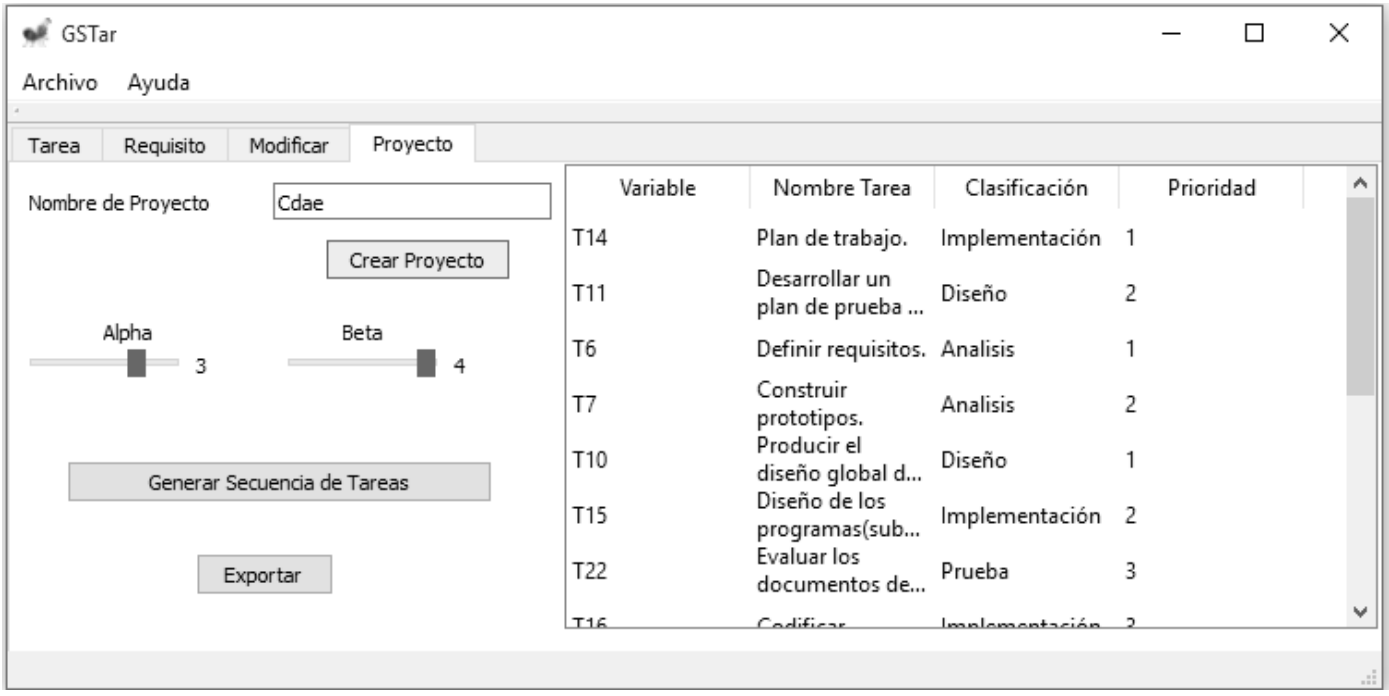

Fig. 22 Interfaz lista de tareas generadas.

La interfaz muestra la lista de tareas generadas.

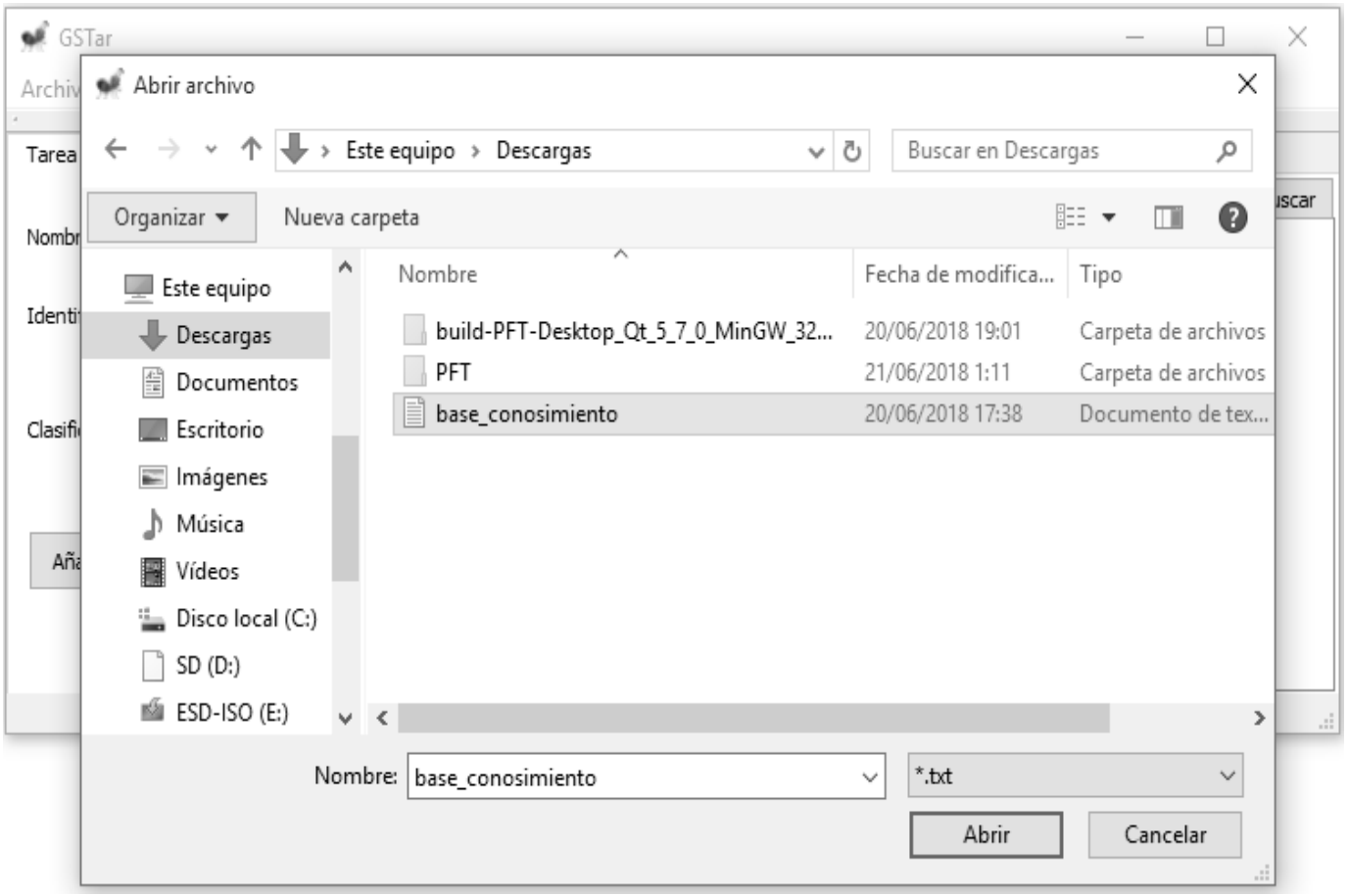

Fig. 23 Interfaz cargar archivo.

En la interfaz se muestra la acción de cargar el archivo que contiene la información que procesara la aplicación. Para esto el usuario da clic en el menú Archivo y luego en la opción Cargar Archivo.

3.5. Pruebas

Las pruebas de software permiten determinar si el producto generado satisface las especificaciones establecidas. Permite detectar la presencia de errores que pudieran generar salidas o comportamientos inapropiados durante su ejecución. Las pruebas de software son importantes porque aseguran el correcto cumplimiento de la funcionalidad del producto, ayudan a ganar confianza, confirman la fiabilidad del uso del software y previenen defectos en producción.

A la aplicación se le realizan pruebas funcionales, las cuales se muestran a continuación divididas en escenarios positivos y negativos, introduciendo datos correctos e incorrectos y visualizando el resultado en cada caso. Las pruebas funcionales están centradas en comprobar que las funcionalidades descritas en el documento de se cumplen con la implementación realizada. A este tipo de pruebas también se les denomina pruebas de comportamiento o de caja negra, debido a que los analistas enfocan su atención a

las respuestas del sistema de acuerdo a los datos de entrada y sus resultados en los datos de salida, los cuales se definen generalmente en los casos de prueba que se crean antes del inicio de las pruebas.

Escenario Añadir Tarea

Escenarios negativos para la acción de añadir tarea.

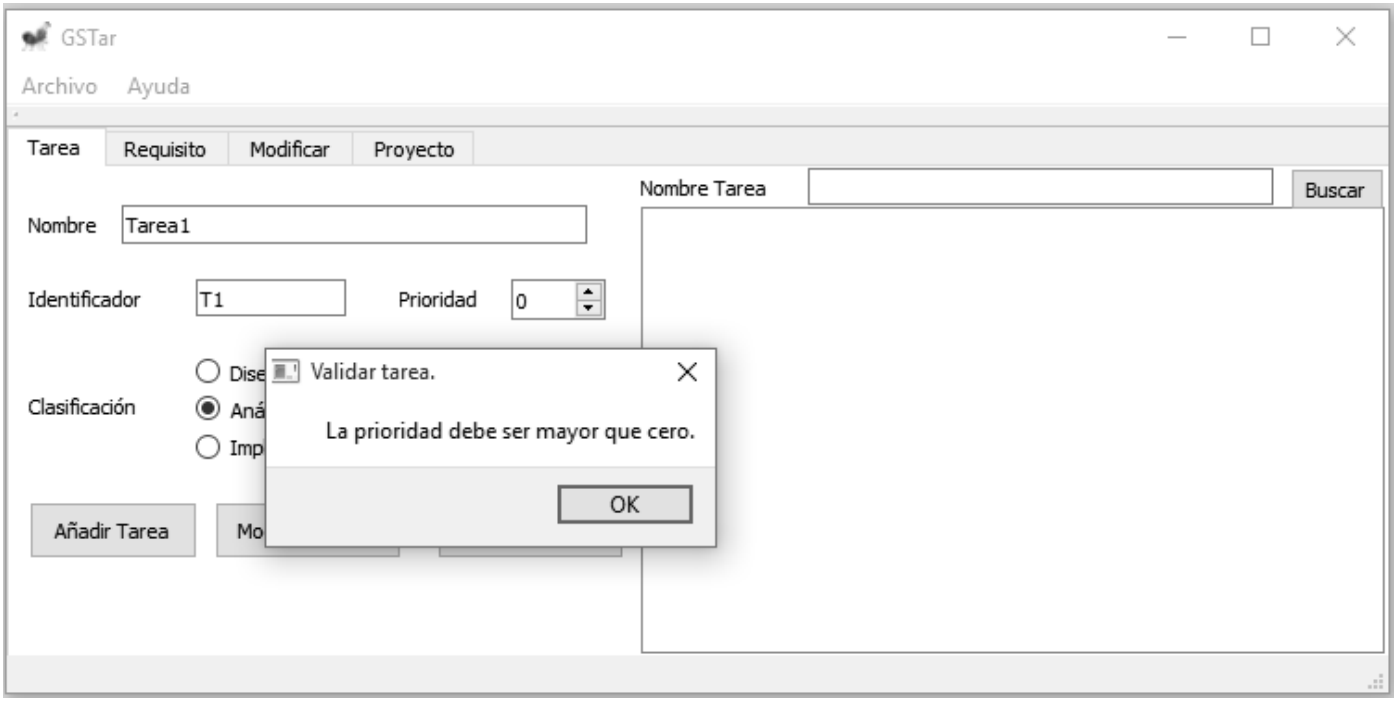

Fig. 24 Escenario negativo de prioridad de tarea.

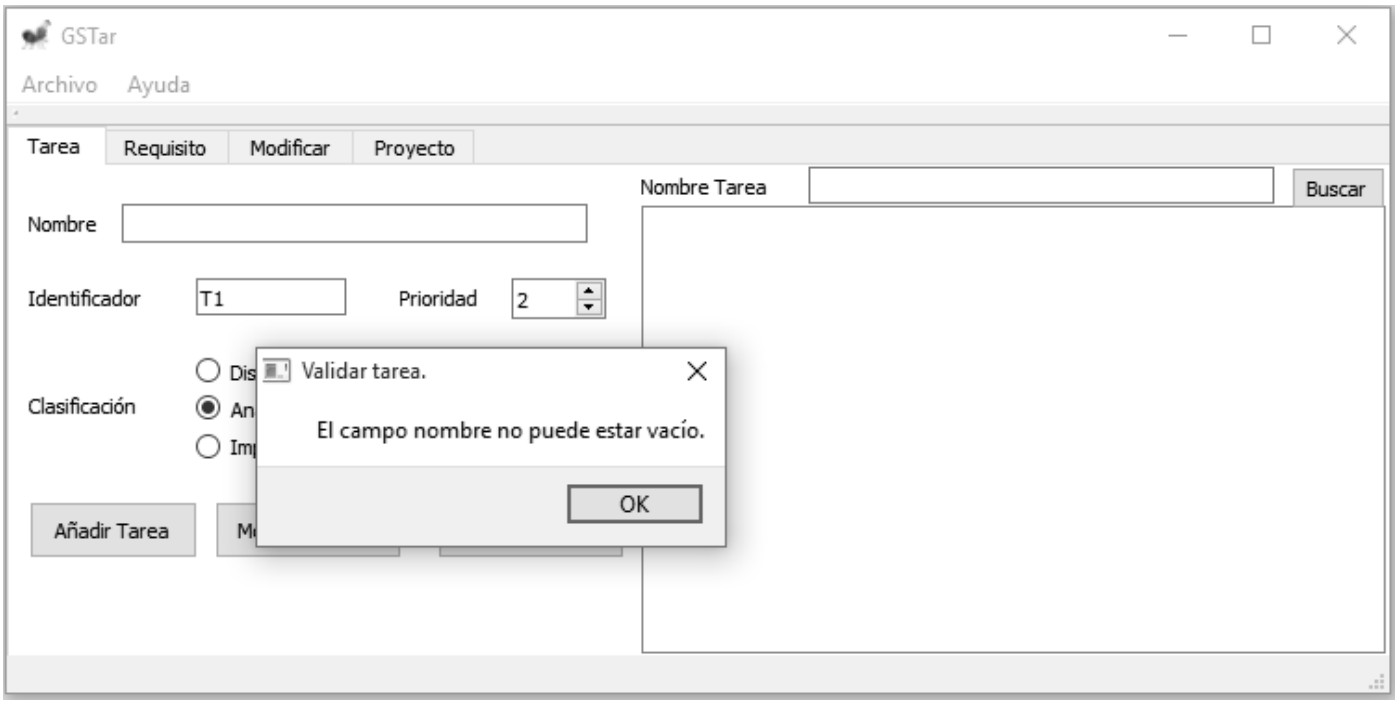

Fig. 25 Escenario negativo campo nombre de tarea.

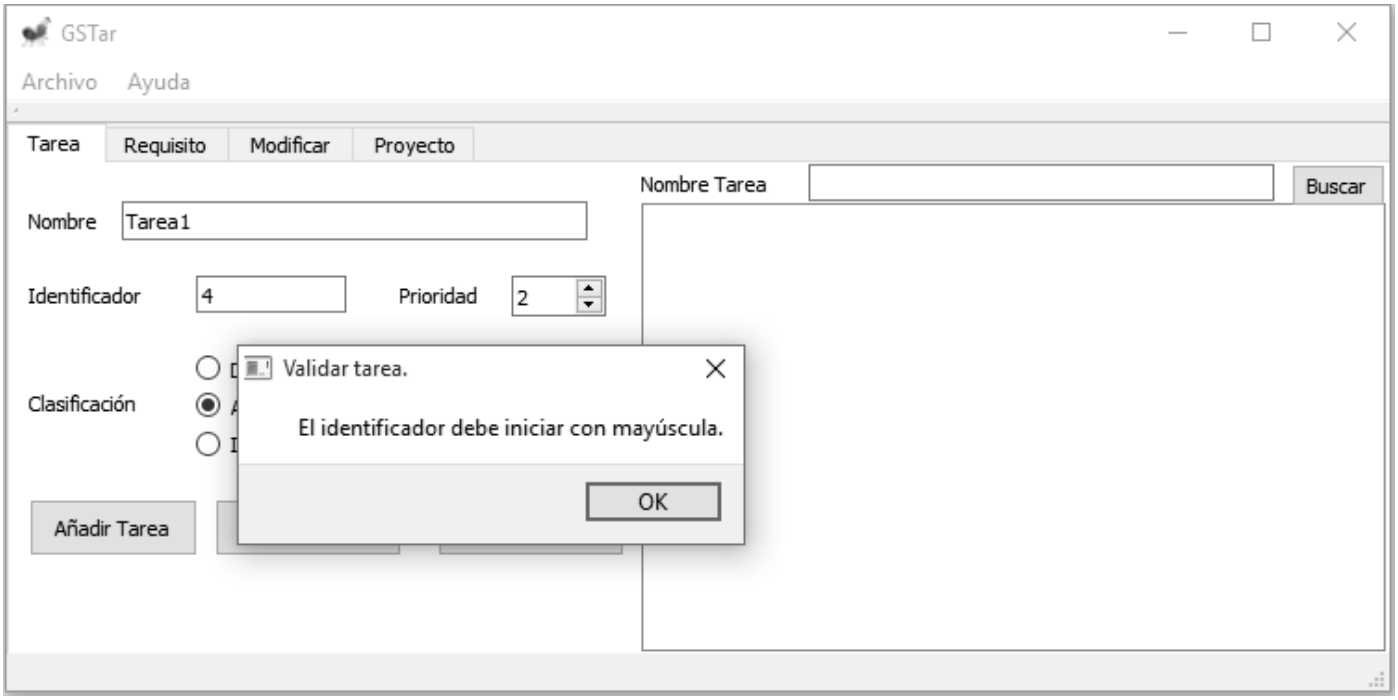

Fig. 26 Escenario negativo de identificador de tarea.

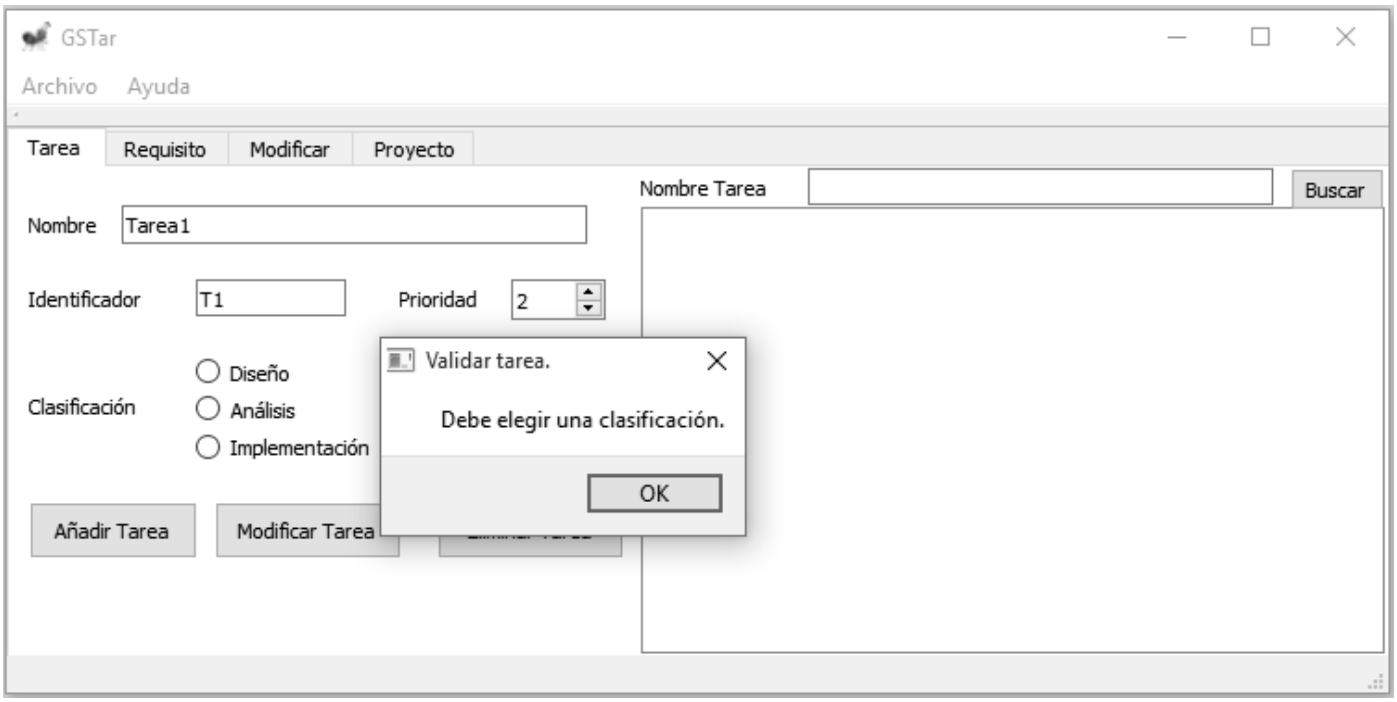

Fig. 27 Escenario negativo de clasificación de tarea.

Escenario positivo de la acción añadir tarea.

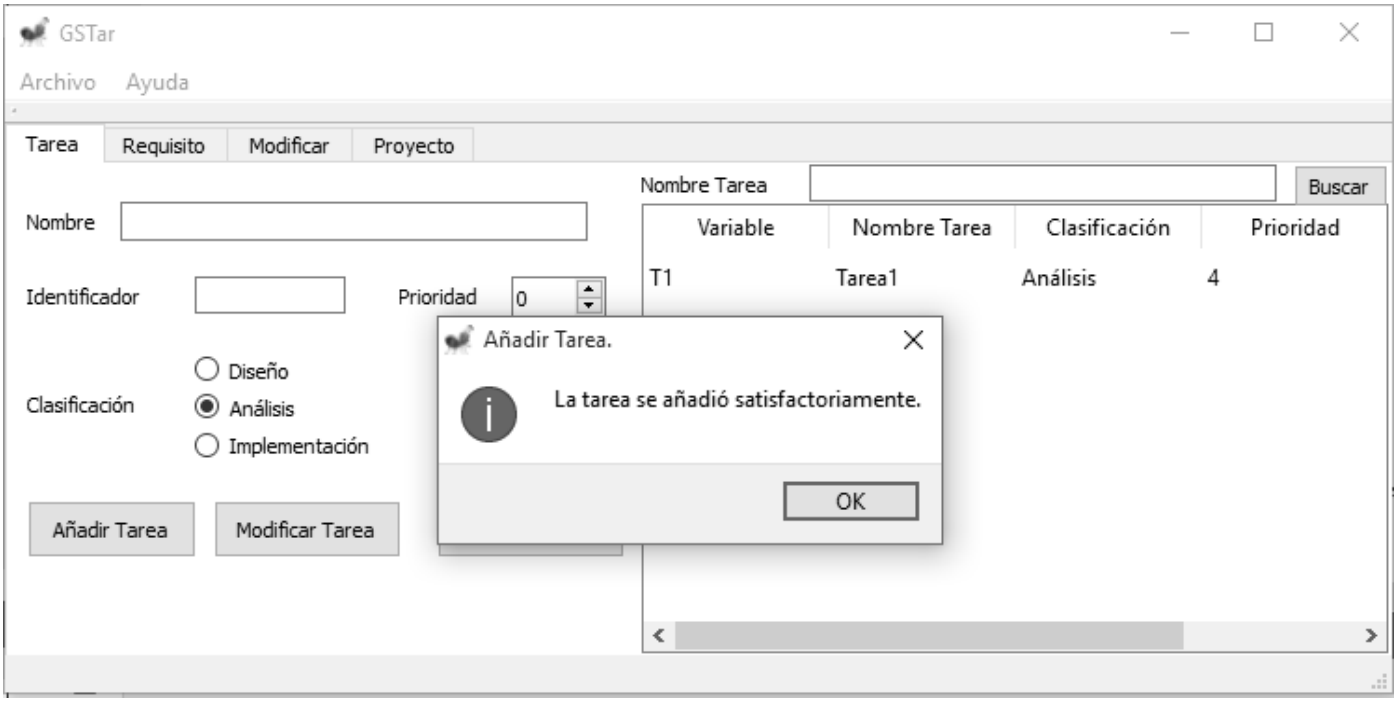

Fig. 28 Escenario positivo añadir tarea.

Escenario modificar tarea.

Escenario negativo de modificar tarea.

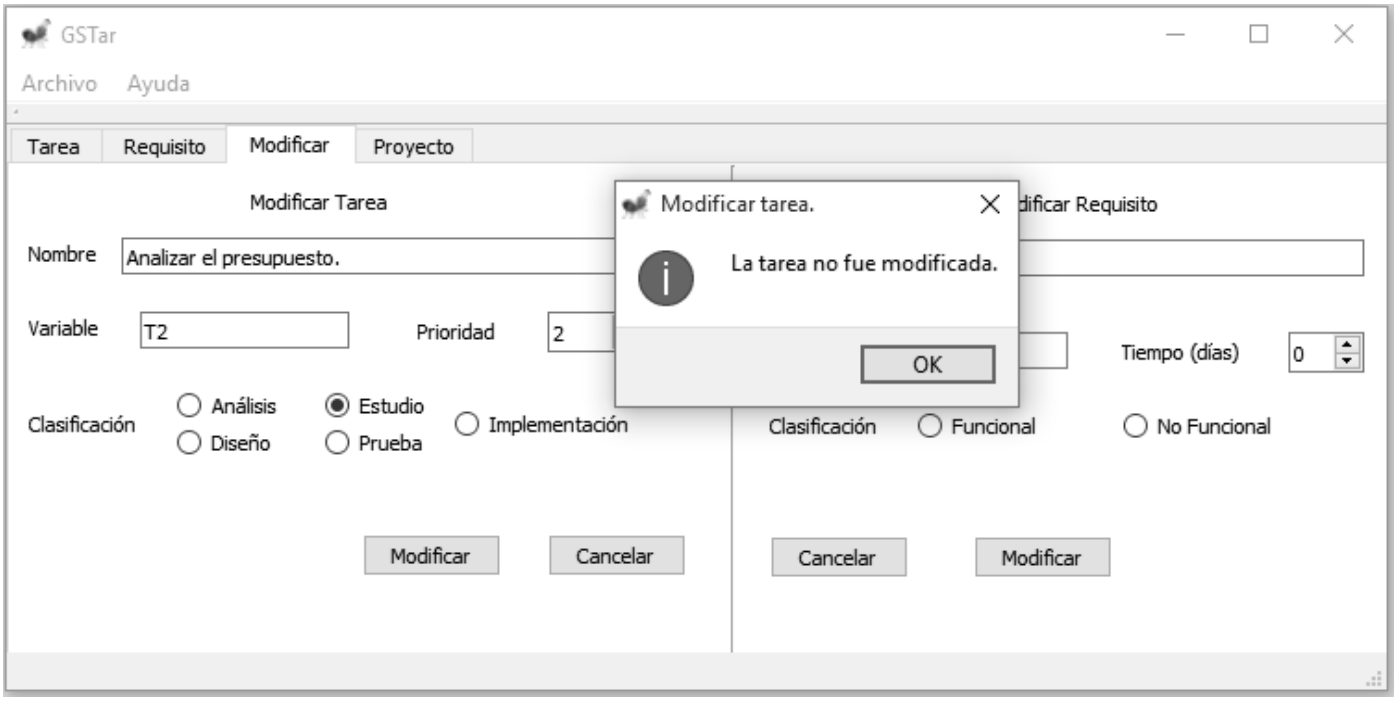

Fig. 29 Escenario negativo de modificar tarea.

Escenario positivo de modificar tarea.

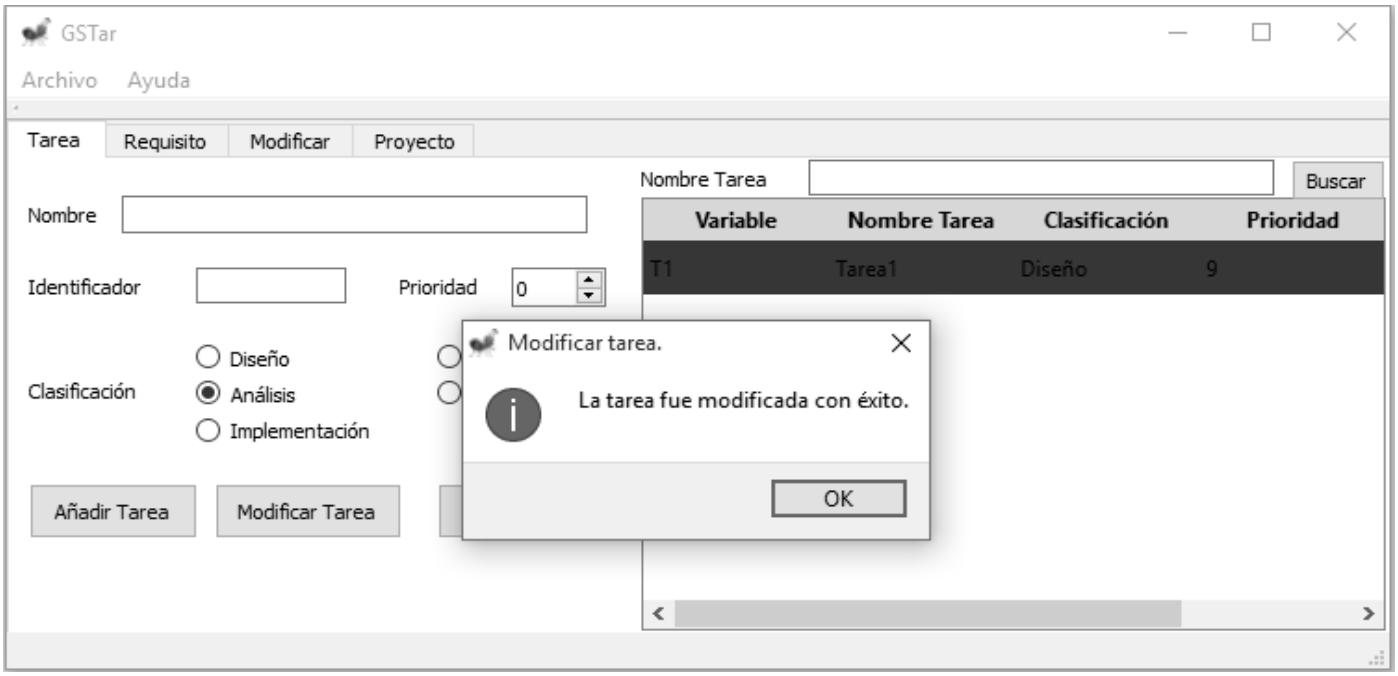

Fig. 30 Escenario positivo modificar tarea.

Escenario eliminar tarea.

Escenario negativo de la acción eliminar tarea.

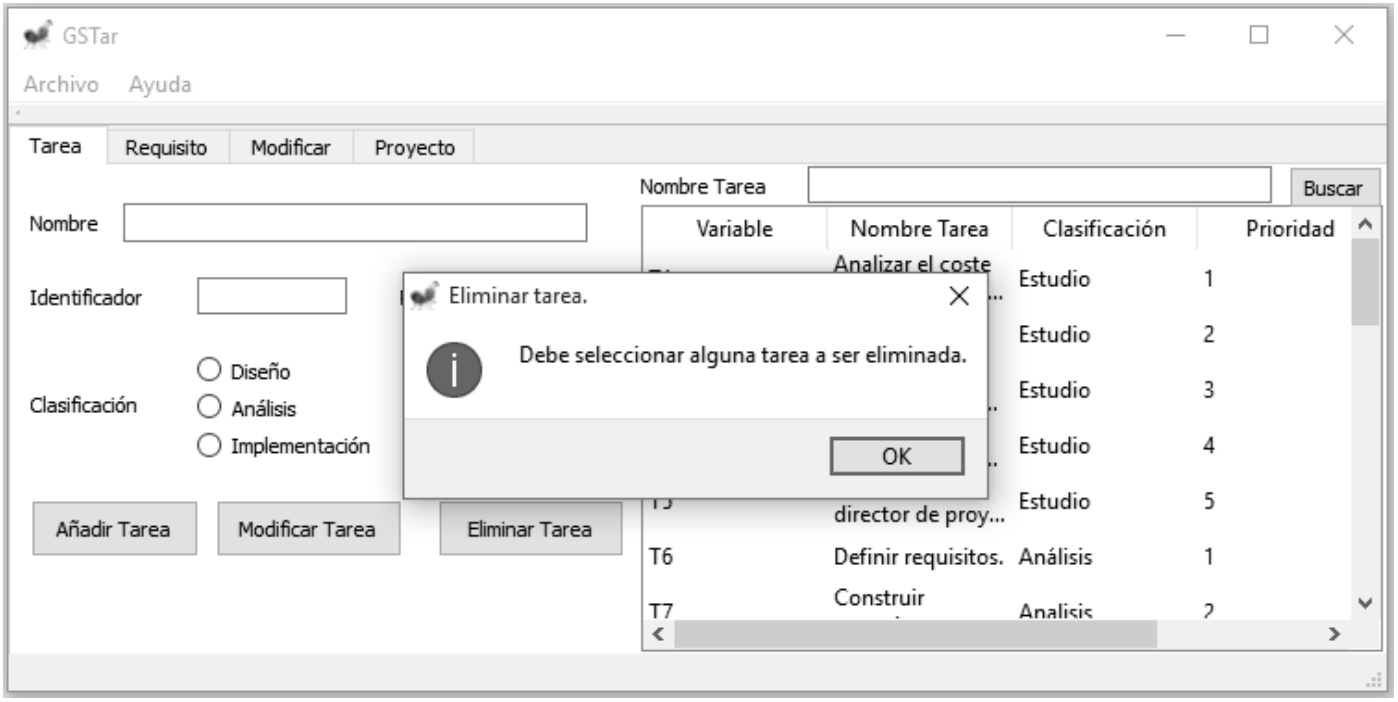

Fig. 31 Escenario negativo eliminar tarea.

Escenario positivo de la acción eliminar tarea.

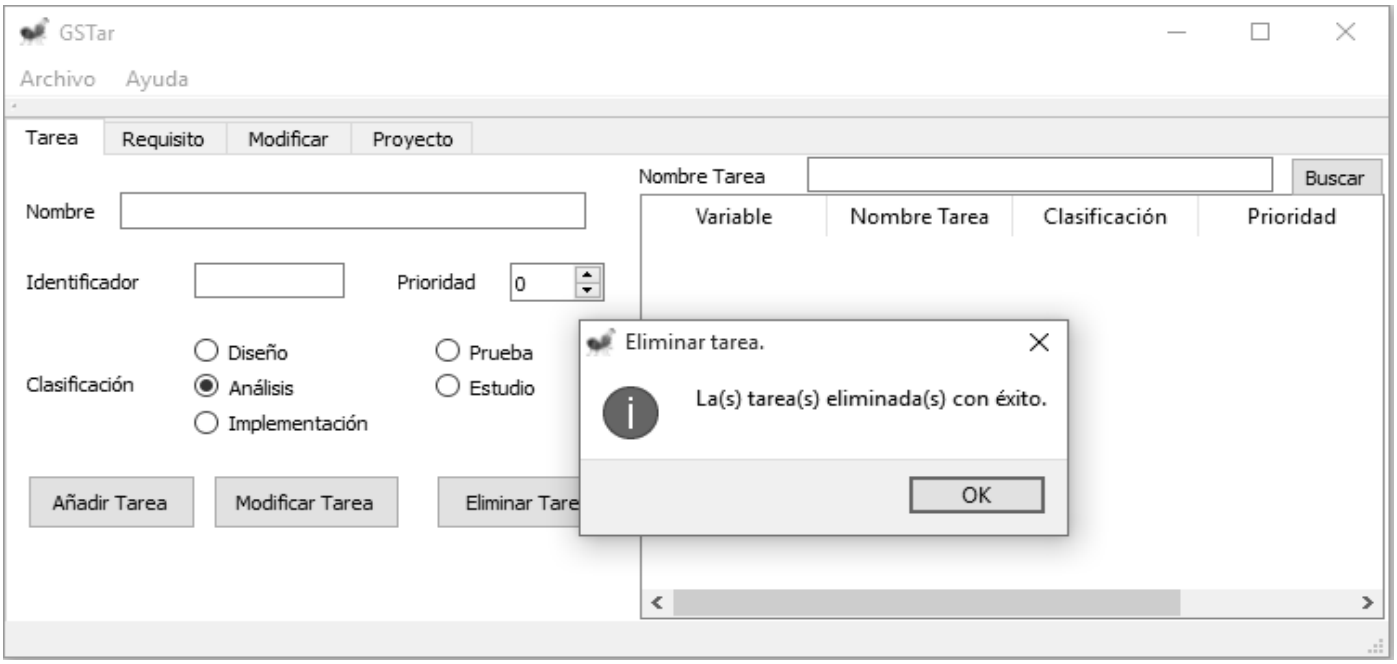

Fig. 32 Escenario positivo eliminar tarea.

Escenarios añadir requisito.

Escenarios negativos de la acción añadir requisito.

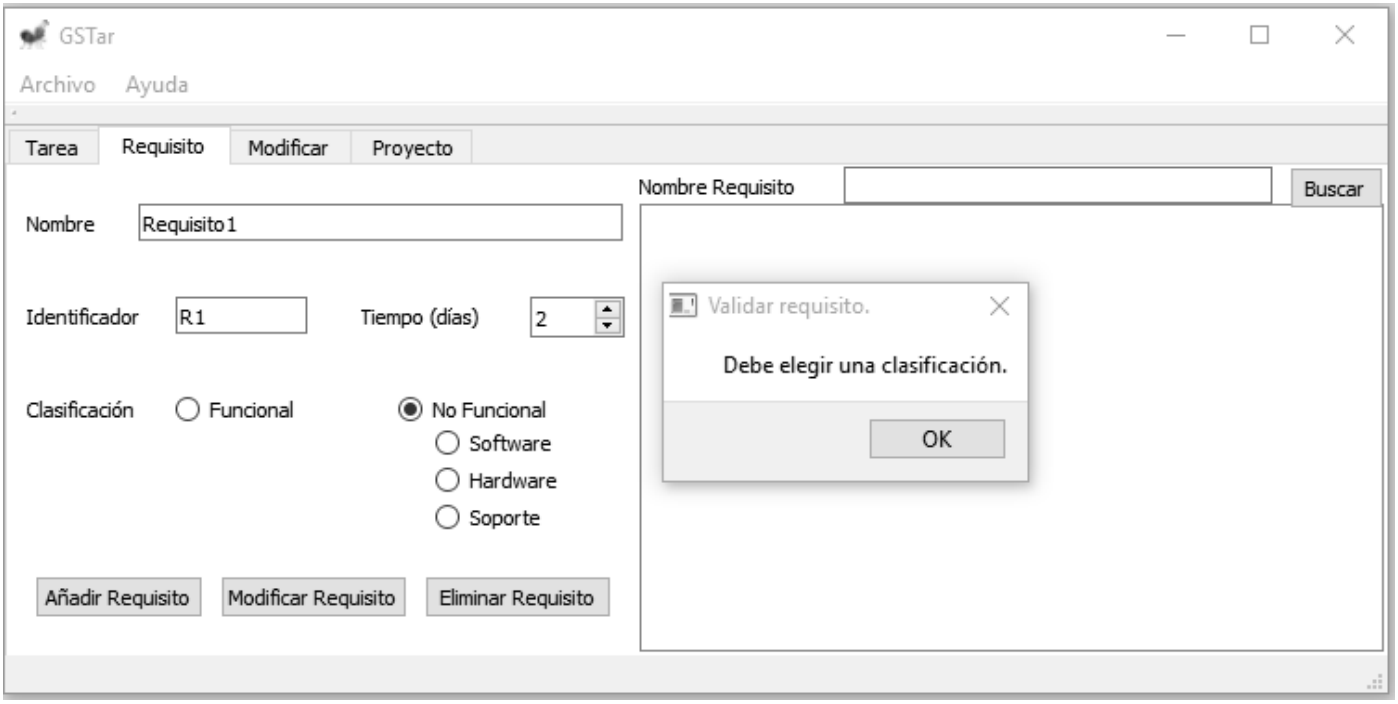

Fig. 33 Escenario negativo clasificación de requisito.

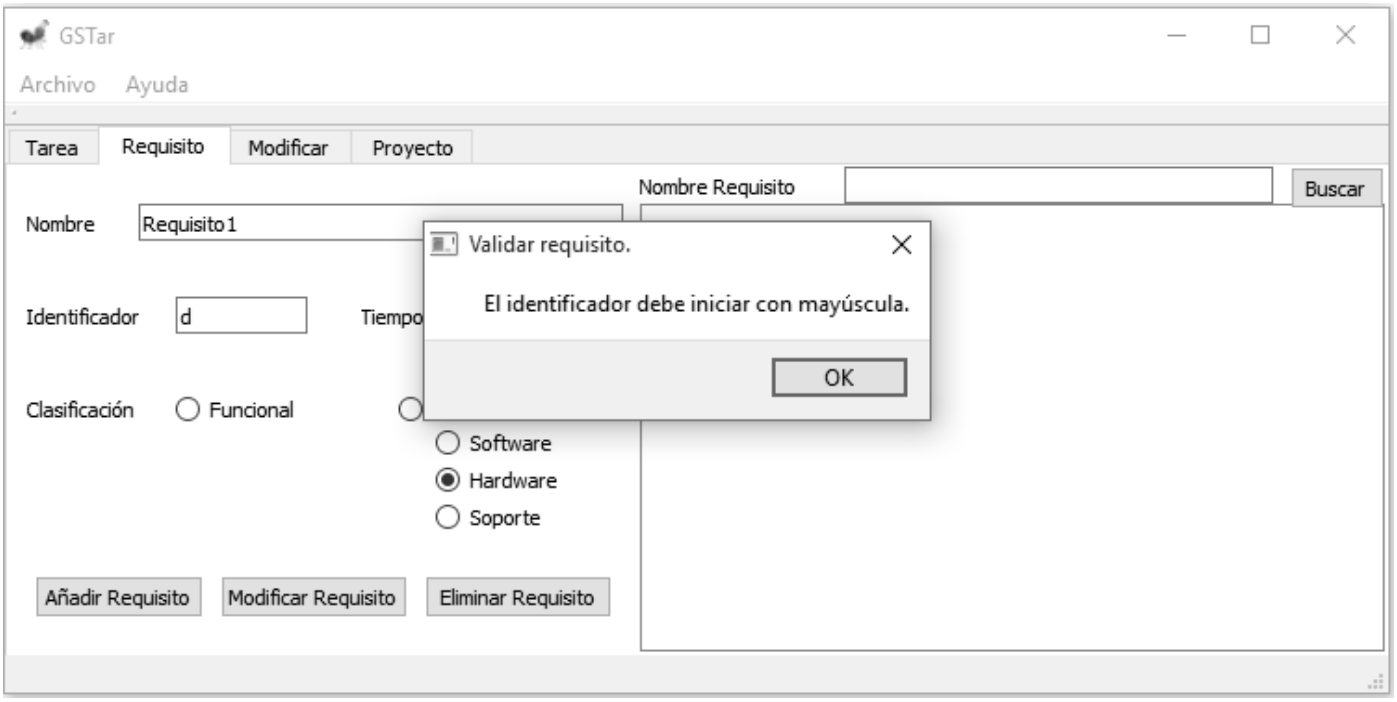

Fig. 34 Escenario negativo identificador de requisito.

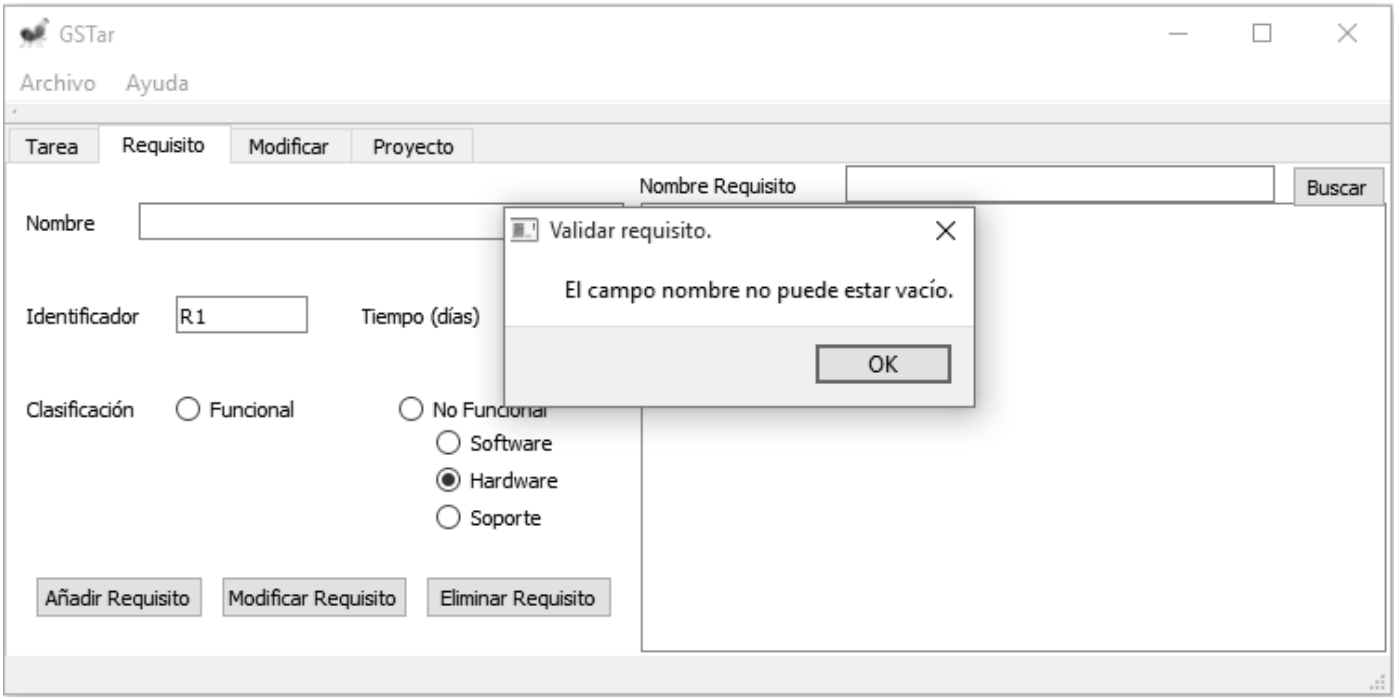

Fig. 35 Escenario negativo nombre de requisito.

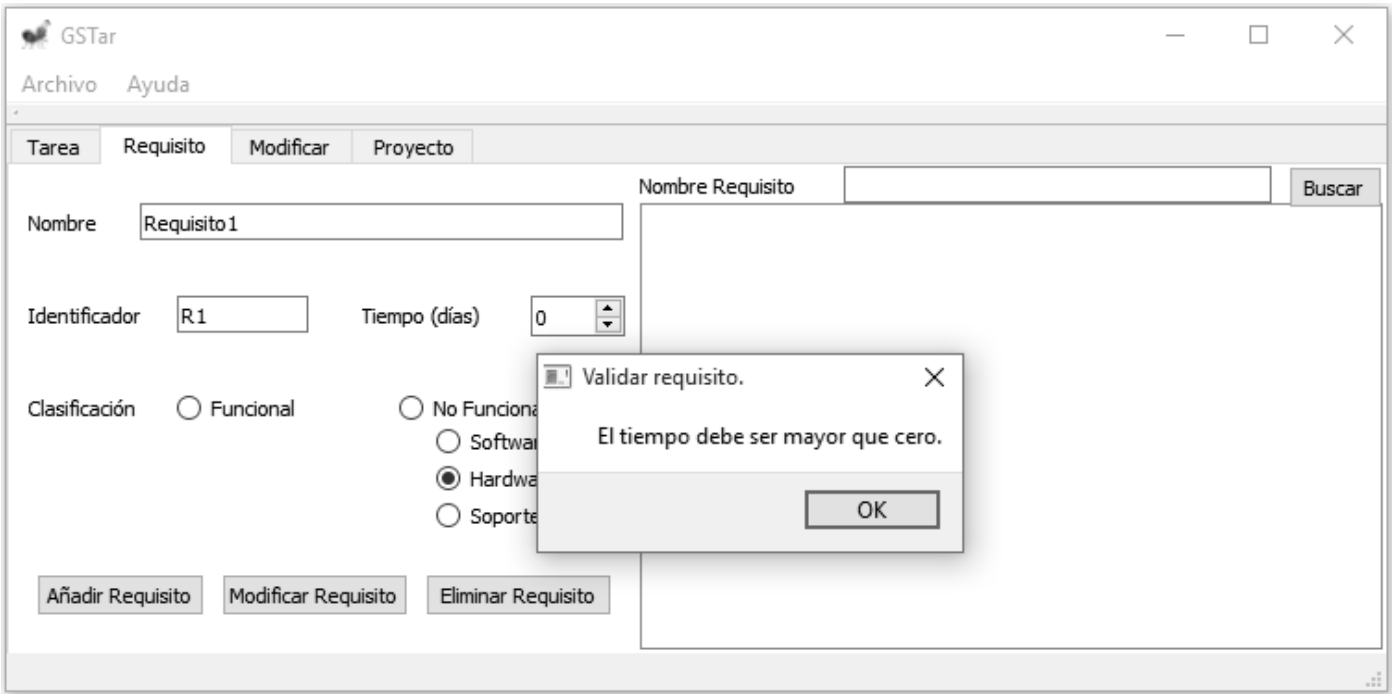

Fig. 36 Escenario negativo tiempo de requisito.

Escenario positivo de la acción añadir requisito.

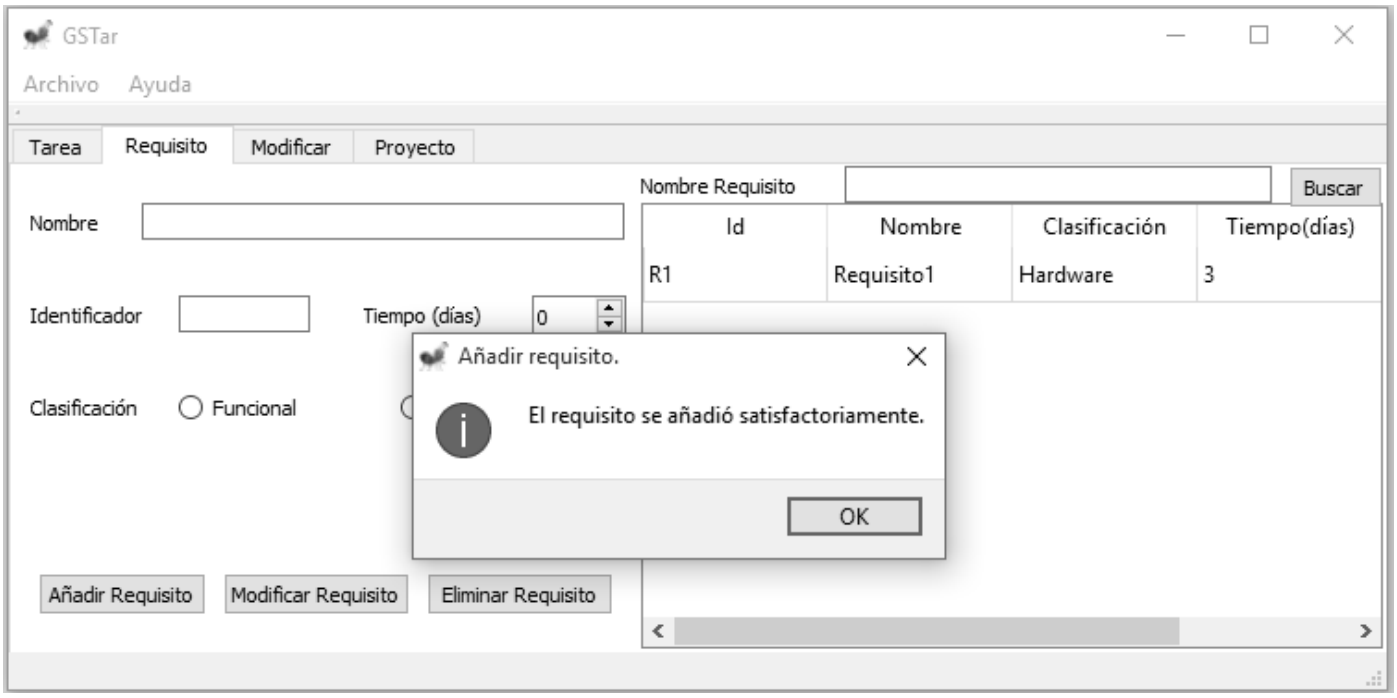

Fig. 37 Escenario positivo de añadir requisito.

Escenario modificar requisito.

Escenario negativo de la acción de modificar requisito.

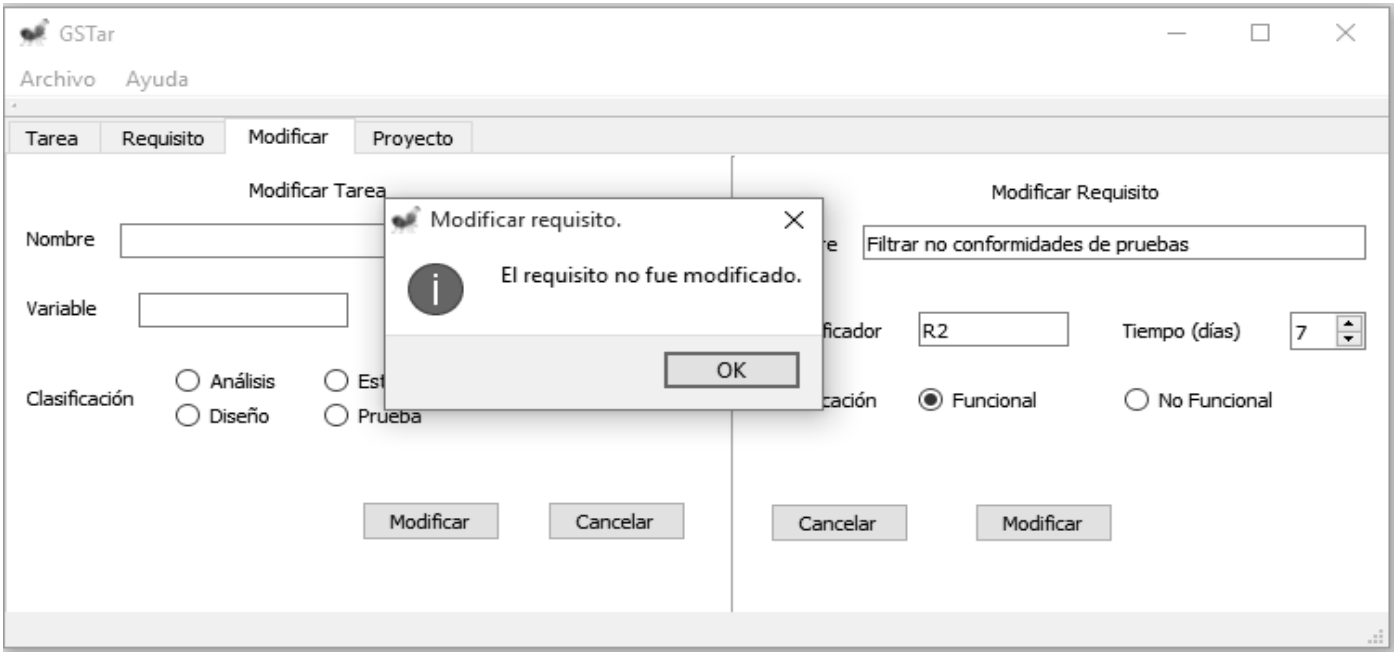

Fig. 38 Escenario negativo de modificar requisito.

Escenario positivo de la acción de modificar requisito.

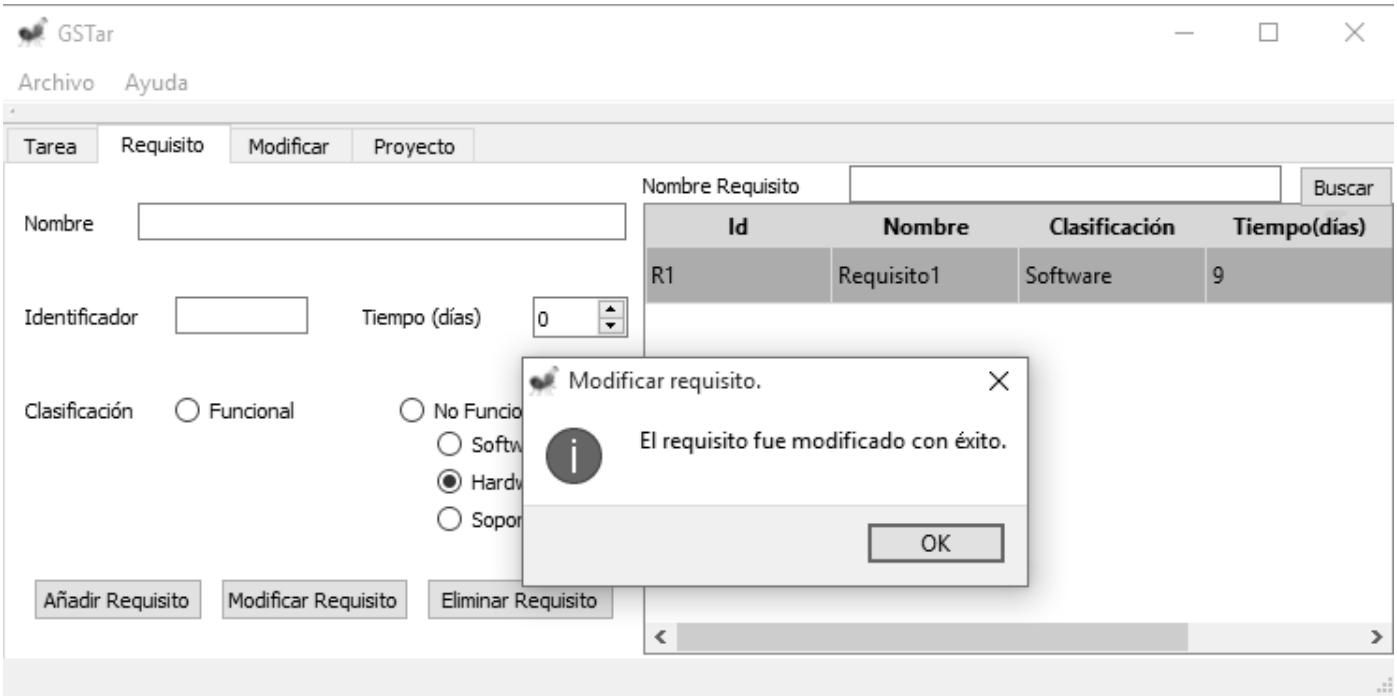

Fig. 39 Escenario positivo de modificar requisito.

Escenario eliminar requisito.

Escenario negativo de la acción de eliminar requisito.

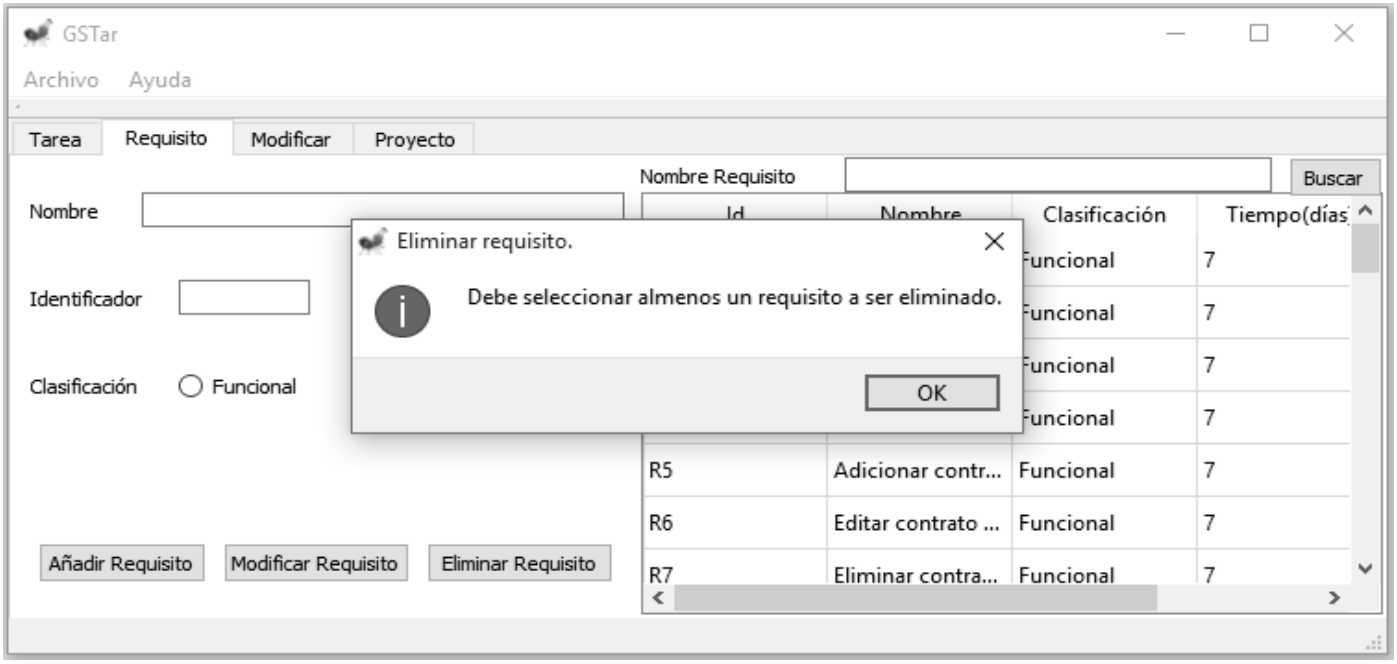

Fig. 40 Escenario negativo eliminar requisito.

Escenario positivo de la acción de eliminar requisito.

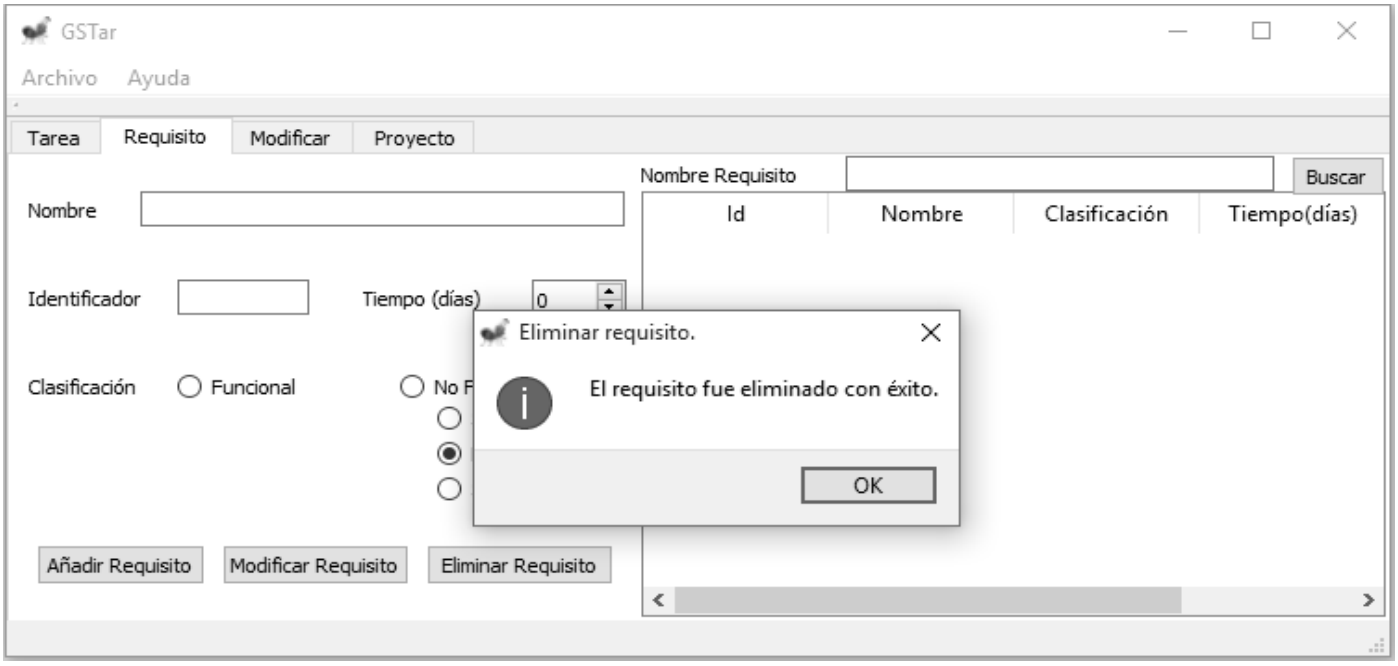

Fig. 41 Escenario positivo eliminar requisito.

Escenario crear proyecto.

Escenario negativo de la acción de crear proyecto.

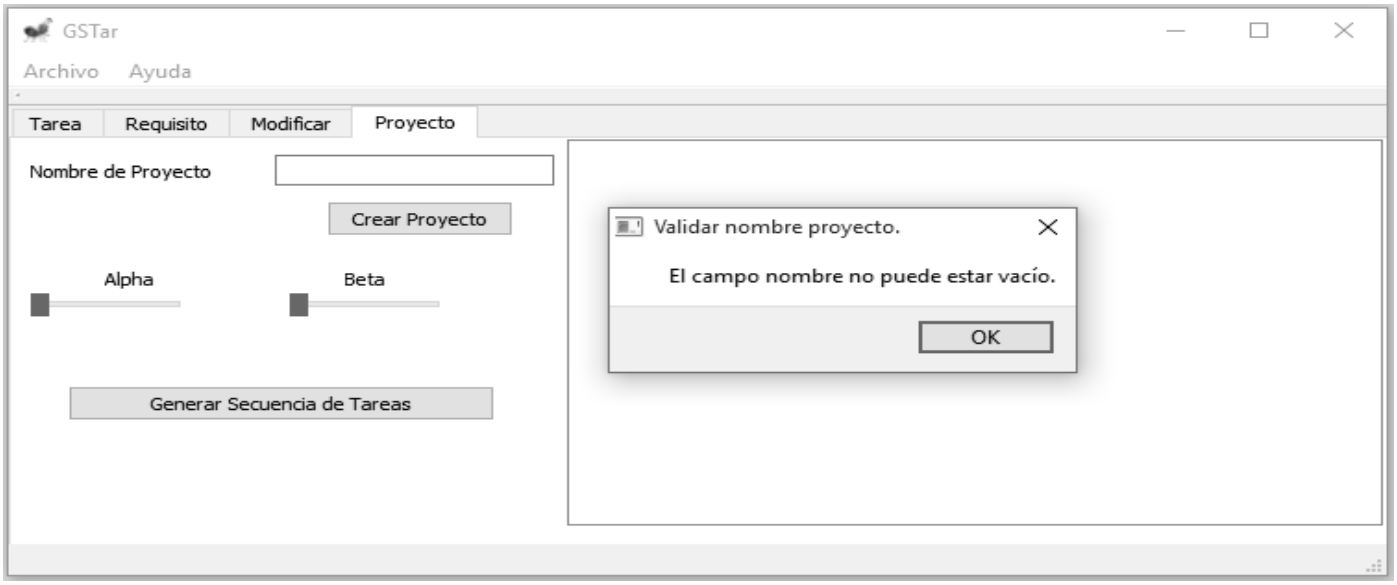

Fig. 42 Escenario negativo crear proyecto.

Escenario positivo de la acción de crear proyecto.

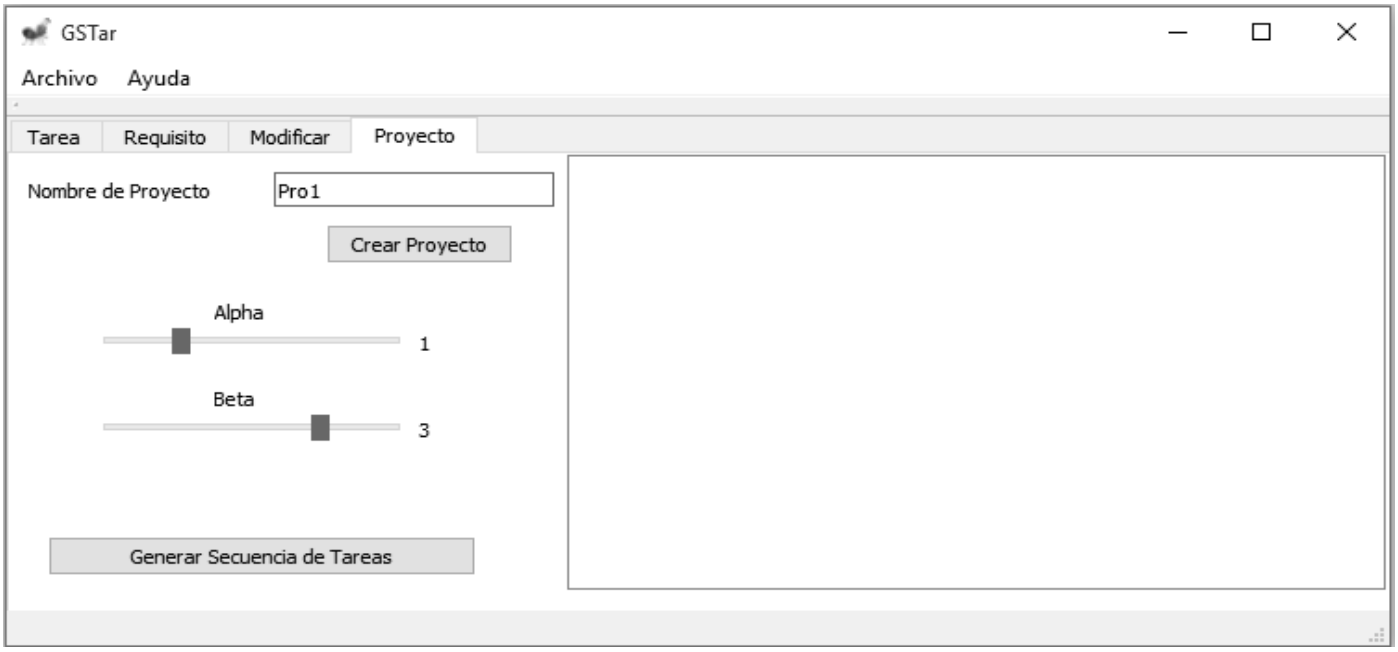

Fig. 43 Escenario positivo crear proyecto.

Escenario generar secuencia de tareas.

Escenario positivo de la acción generar secuencia de tareas.

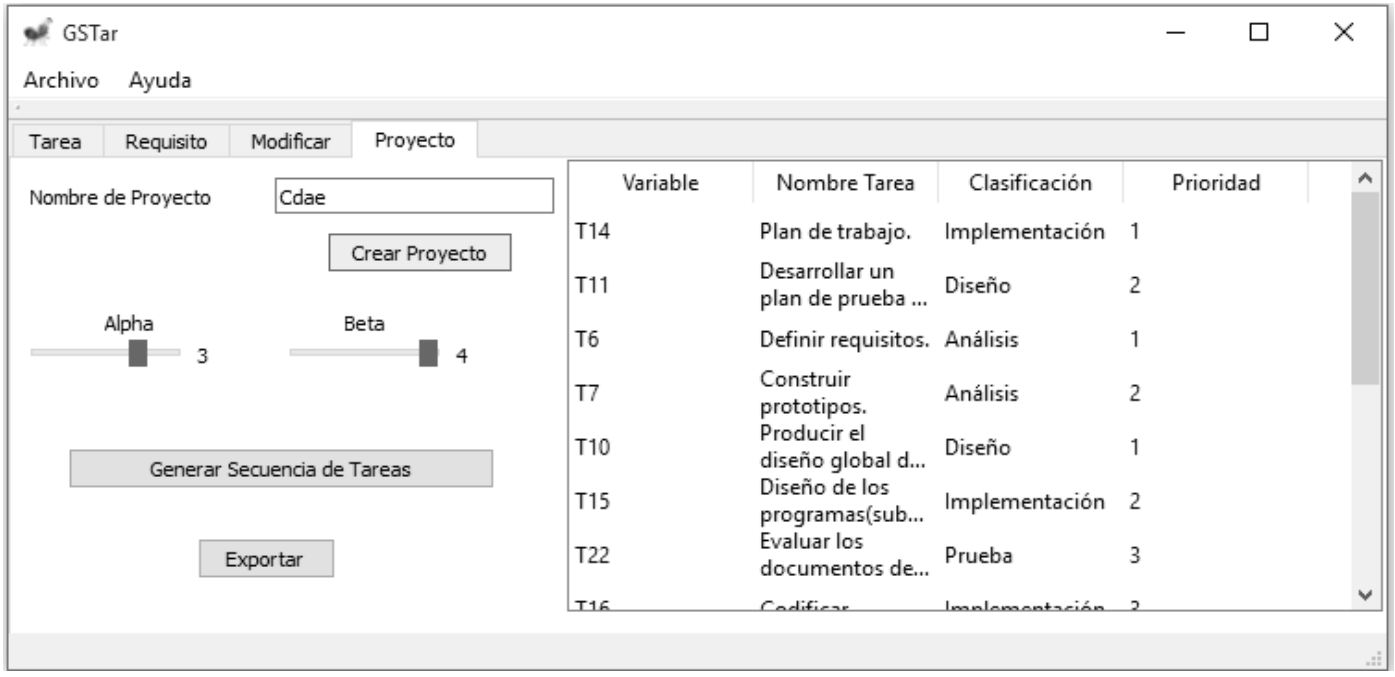

Fig. 44 Escenario positivo generar secuencia de tareas.

Escenario exportar secuencia de tareas.

Escenario positivo de la acción exportar secuencia de tareas.

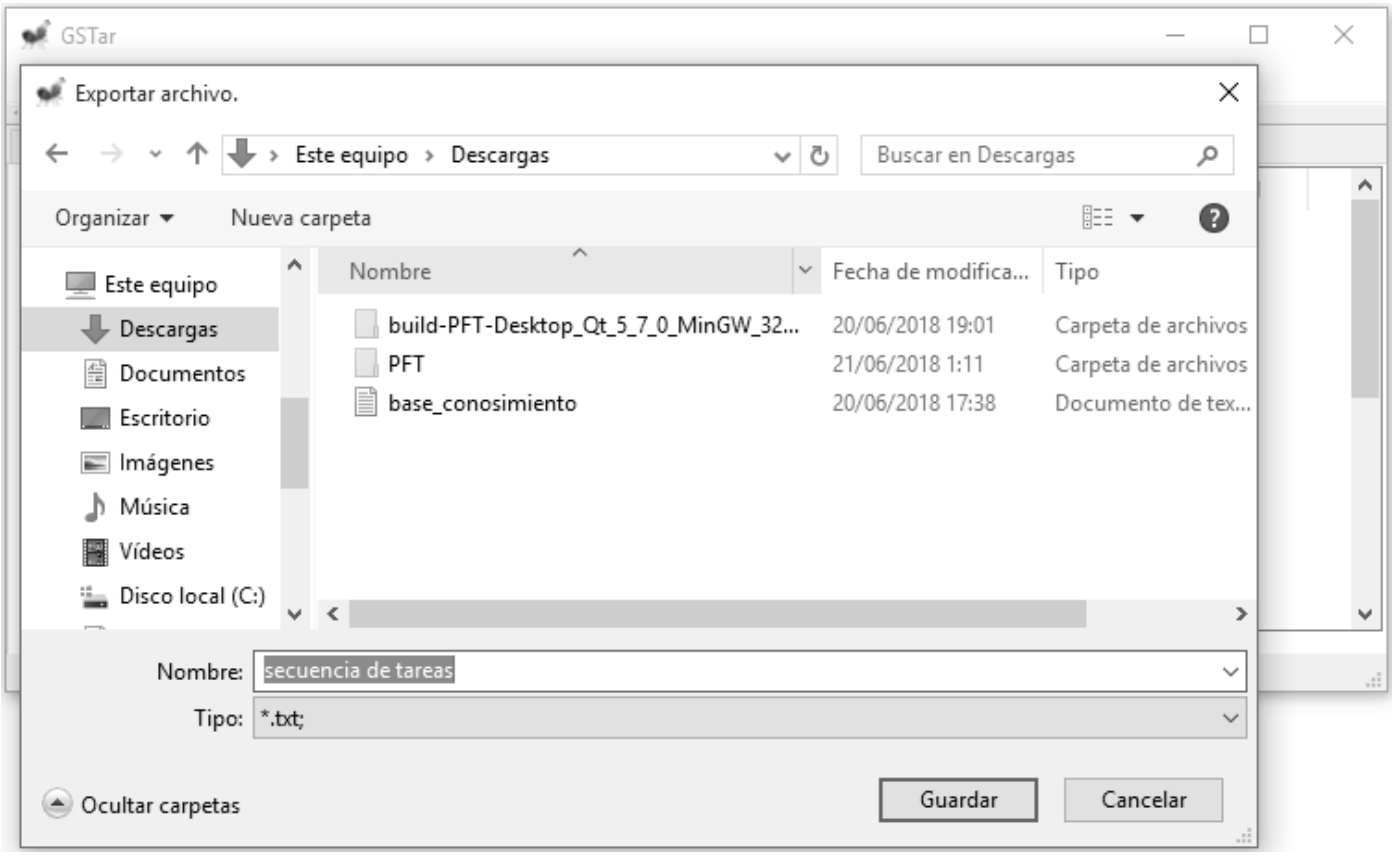

Fig. 45 Escenario positivo de la acción exportar secuencia de tareas.

Conclusiones

Concluida la presente investigación se obtiene una aplicación que permite generar y optimizar secuencias de tareas en función de un conjunto de requisitos de proyectos de software cumpliendo así el objetivo trazado y se arriban a las siguientes conclusiones:

- Mediante consultas y entrevistas a expertos en el tema de investigación se obtuvieron un conjunto de tareas y requisitos, de los cuales se tomó una muestra de 24 tareas y 32 requisitos, lo que permitió el procesamiento de los datos y encontrar como resultados una secuencia de tareas optimizada.
- A partir de las tareas y requisitos se realizó un análisis en cuanto a las relaciones existente entre los requisitos y las tareas, lo que contribuyó a una representación más clara y precisa de la dependencia entre estos.
- Se obtuvo un grafo ponderado que permitió la aplicación del ACH en la búsqueda de una secuencia de tareas optimizada para lograr una mejor planeación en cuanto a la implementación de los requisitos de software en un proyecto.
- La validación de la aplicación de la propuesta de solución por medio de pruebas de aceptación demostró su correcto funcionamiento y el cumplimiento de los requisitos establecidos por el cliente.

Recomendaciones

- Extender el análisis al comportamiento de tareas en función de los recursos que intervienen en el proyecto, vinculados al análisis de recursos, costo y calidad de Software.
- Incorporar a la aplicación un módulo de validación semántica y sintáctica de los nombres de los requisitos y las tareas.
- Implementar la funcionalidad de mostrar al usuario las tareas que no se incorporaron en la secuencia de tareas optimizadas generadas.
- Agregar a la aplicación la posibilidad de importar y exportar de un archivo .csv.

Bibliografía

¡Error! No se le ha dado un nombre al marcador. Aparicio, D. 2012. Aplicación de los algoritmos de hormigas para la resolución de un RALBP.
Barcelona.

Bonabeau, E and Dorigo, M.1999. "Swarm Intelligence: From Natural to Artificial Systems", Oxford University Press US.

Bello, R. 2010. Analysis of the efficacy of a Two-Stage methodology for ant colony optimization: Case of study with TSP and QAP.https://www.sciencedirect.com/science/article/pii/S0957417410001089

Buntara, G., Takahiro, H., Aylie, H., Alisjahbana, S. y Asád, S. (2017). Evolutionary ACO algorithms for truss optimization problems. Procedía Engineering, 171. doi: 10.1 O 16/j. proeng.2017. O 1.467

Brossard, Yulie. 2009.Diagnóstico de Enfermedades de Transmisión Sexual mediante técnicas de Inteligencia Artificial. Habana. Cuba.

Carrizo Danteand Quintanilla, Ivan *Adequacy profile of software requirements elicitation techniques 2016.Revista de ingenieria. Chile,*

Collazos, C. (2013). Rediseño del sistema productivo utilizando técnicas de distribución
de planta. Manizales. Obtenido de http://www.bdigital.unal.edu.co/12157/1/8912504.2013.pdf

Dorigo, M and Maniezzo, V.1991. "Positive Feedback as a Search Strategy", ipartimento di Elettronica, Politecnico di Milano.

IBARRA, María de los Angeles. 2006. *Procesamiento Analítico en Línea.* Universidad Nacional del Nordeste. Argentina ..

J. Cuadra,D. y Isasi, P. (2017). Extending ACO for fast path search in huge graphs and social networks

Millet, R., Beyris, M., y Rosales, M. (2011). Colonia de hormigas aplicada a la teoría de grafos. Cuba.

Ortega, A., y Ana, S. 2011. Un Algoritmo Híbrido Basado en Colonias de Hormigas
para la Resolución de Problemas. XIII Jornadas de ASEPUMA, 3. Pérez, I.

Pérez, S. (2013). Implementación de un algoritmo basado en Colonias de Hormigas para la optimización de funciones con datos mezclados. Las Villas.

Rodríguez, C. D. y Melgarejo, M. A.2015. Arquitectura FPGA para simulación de aprovisionamiento de alimentos en colonias de hormigas artificiales. ‘En: Ingeniería, Vol. 20, No. 2, pp. 245–260

Ángel, Rafael.2017. Algoritmo perfeccionado de hormigas artificiales. Agosto 2017. https://www.lawebdelprogramador.com/codigo/Java/4102-Algoritmo-perfeccionado-de-hormigas-artificiales.html

Brownlee, Jason. 2015. Clever Algorithms: Nature-Inspired Programming Recipes. http://www.cleveralgorithms.com/nature-inspired/swarm/ant_system.html

Fernández, Alejandra., Rivera, Mónica y Acosta, Evelin. 2013. Laboratorio de sistemas adaptativos. http://2013sistemasadaptativos.blogspot.com/2013/09/algoritmo-de-optimizacion-por-colonia.html

Sancho Caparrini, Fernando. 2016. Algoritmos de hormigas y el problema del viajante. http://www.cs.us.es/~fsancho/?e=71

Alejandro Arito, Franco Luis., Leguizamón, Dr. Guillermo y Errecalde, Dr. Marcelo. 2010. Algoritmos de Optimización basados en Colonias de Hormigas aplicados al Problema de Asignación Cuadrática y otros problemas relacionados.

Rodríguez García, Jesús. 2010. Análisis de algoritmos basados en colonia de hormigas en problemas de camino mínimo.

Atehortúa, Ant Colony Optimization: Generalities and Study of Ant System Algorithm and a Job Shop Application, National University of Colombia, Bogotá, Colombia 2012.

M. Gendreau, J. Y. Potvin, Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 146, Springer, 2010.

M. H. Ha, N. Bostel, A. Langevin, L. M. Rousseau, An Exact Algorithm and a Metaheuristic for the Generalized Vehicle Routing Problem, Computers & Operations Research, vol. 43, Elsevier Publishing, 2014.

Soifer, Alexis. 2015.Algoritmos de Colonia de Hormigas para el Problema del Viajante de Comercio por Familias y para el Problema de Ruteo de Vehículos por Familias.

Anexos

Anexo 1: Ayuda de la aplicación.

Anexo 2 Código para cargar una tarea a la interfaz modificar.

```
//pasa la tarea que vas a modificar
void MainWindow::Pasar_Tarea_A_Modificar(Tarea * edit_tarea, int pos)
{
ui->lineEditnombremodificar->setText(edit_tarea->getTexto_tarea());
ui->lineEditvariablemodificar->setText(edit_tarea->getId_tarea());
if(edit_tarea->getClasificacion().compare("Analisis")==0)
ui->radioButtonanalisismodificar->setChecked(true);
else
if(edit_tarea->getClasificacion().compare("Diseño")==0)
ui->radioButtondisennomodificar->setChecked(true);
else
if(edit_tarea->getClasificacion().compare("Implementación")==0)
ui->radioButtondesarrollomodificar->setChecked(true);
else
if(edit_tarea->getClasificacion().compare("Prueba")==0)
ui->radioButtonpruebamodificar->setChecked(true);
else
```

```
if(edit_tarea->getClasificacion().compare("Estudio")==0)
ui->radioButtonestudiomodificar->setChecked(true);

ui->spinBoxprioridadmodificar->setValue(edit_tarea->getPrioridad());
ui->labelmodificartarea->setAccessibleName(QVariant(pos).toString());
}
```

Anexo 3: Código para añadir las tareas a partir del fichero.

```
// crear tareas a partir del fichero
void MainWindow::LlenarListaTareaFromArchivo(QString cad)
{
    QString variable, clasificacion, nombre, prioridad;
    variable = DescomponerAtributosTarea(cad);
    clasificacion = DescomponerAtributosTarea(cad);
    prioridad = DescomponerAtributosTarea(cad);
    nombre = cad;
    task = new Tarea(variable,clasificacion,prioridad.toInt(),nombre);
    proyecto->getList_tarea()->append(task);
}
```

Anexo 4: Código para añadir una tarea.

```
// Añadir una tarea
void MainWindow::on_pushButton_clicked()
{
    QString id_tarea = ui->lineEditvariabletarea->text();
    QString clasificacion = Clasificacion_Tarea();
    int prioridad = ui->spinBoxofprioridad->text().toInt();
     QString texto_tarea = ui->lineEditofnombretarea->text();
     if(clasificacion=="" ||  prioridad<=0  || texto_tarea.size()<=0 || !ValidarIdTarea(id_tarea)){
        if(clasificacion==""){
            QMessageBox m;
            m.setWindowTitle("Validar tarea.");
            m.setText("Debe elegir una clasificación.");
            m.exec();
        }
            if( prioridad<=0){
                QMessageBox m;
                m.setWindowTitle("Validar tarea.");
                m.setText("La prioridad debe ser mayor que cero.");
                m.exec();
            }
                if(texto_tarea.size()<=0){
                    QMessageBox m;
                    m.setWindowTitle("Validar tarea.");
```

```
                m.setText("El campo nombre no puede estar vacio.");
                m.exec();
            }
    } else
    {
        if(proycreado == false){
            task = new Tarea(id_tarea, clasificacion, prioridad, texto_tarea);
           proyecto->getList_tarea()->append(task);
           LlenarTablaTarea(ui->tableWidget,proyecto->getList_tarea());
           ui->lineEditvariabletarea->setText("");
           ui->spinBoxofprioridad->setValue(0);
           ui->lineEditofnombretarea->setText("");
           ui->radioButton->setChecked(false);
           ui->radioButton_2->setChecked(false);
           ui->radioButton_5->setChecked(false);
           ui->radioButton_4->setChecked(false);
           ui->radioButton_3->setChecked(false);
           QMessageBox::information(this, tr("Añadir Tarea."),
                                    tr("La tarea se añadió satisfactoriamente.\n"),
                                    QMessageBox::Ok);
        } else {

     classe_Control->Adicionar_Tarea(id_tarea,prioridad,clasificacion,texto_tarea);
     LlenarTablaTarea(ui->tableWidget,classe_Control->getProy_Prueba()->getList_tarea());
       ui->lineEditvariabletarea->setText("");
       ui->spinBoxofprioridad->setValue(0);
       ui->lineEditofnombretarea->setText("");
       ui->radioButton->setChecked(false);
       ui->radioButton_2->setChecked(false);
       ui->radioButton_5->setChecked(false);
       ui->radioButton_4->setChecked(false);
       ui->radioButton_3->setChecked(false);
       QMessageBox::information(this, tr("Añadir Tarea."),
                                tr("La tarea se añadió satisfactoriamente.\n"),
                                QMessageBox::Ok);
     }
}
}
```

Anexo 5: Código para modificar una tarea.

```
// Modificar o no una tarea
void MainWindow::on_pushButton_4_clicked()
{
    QString pos = ui->labelmodificartarea->accessibleName();
    QString variable = ui->lineEditvariablemodificar->text();
    QString nombre = ui->lineEditnombremodificar->text();
    QString clasificacion = ClasificacionModificar();
    int prioridad =  QVariant(ui->spinBoxprioridadmodificar->text()).toInt();
```

```
    task = new Tarea(variable,clasificacion,prioridad,nombre);

    if(proycreado){
       if(Tarea_Is_The_Same(task, classe_Control->getProy_Prueba()->getList_tarea()->at(pos.toInt()))){
          QMessageBox::information(this, tr("Modificar tarea."),
                                   tr("La tarea no fue modificada.\n"),
                                   QMessageBox::Ok);
       }
       else
       {
          if(clasificacion=="" ||  prioridad<=0  || nombre.size()<=0){
            if(clasificacion==""){
               QMessageBox m;
               m.setWindowTitle("Validar tarea.");
               m.setText("Debe elegir una clasificación.");
               m.exec();
            }
               if( prioridad<=0){
                  QMessageBox m;
                  m.setWindowTitle("Validar tarea.");
                  m.setText("La prioridad debe ser mayor que cero.");
                  m.exec();
               }
                  if(nombre.size()<=0){
                     QMessageBox m;
                     m.setWindowTitle("Validar tarea.");
                     m.setText("El campo nombre no puede estar vacio.");
                     m.exec();
                  }
}
          else {

             classe_Control->getProy_Prueba()->getList_tarea()->replace(pos.toInt(),task);
             LlenarTablaTarea(ui->tableWidget,classe_Control->getProy_Prueba()->getList_tarea());
             Tarea_X_defecto();
             ui->lineEditvariablemodificar->setText("");
             ui->lineEditnombremodificar->setText("");
             ui->radioButtonanalisismodificar->setChecked(false);
             ui->radioButtondesarrollomodificar->setChecked(false);
             ui->radioButtondisennomodificar->setChecked(false);
             ui->radioButtonestudiomodificar->setChecked(false);
             ui->radioButtonpruebamodificar->setChecked(false);
             ui->spinBoxprioridadmodificar->setValue(0);
             QMessageBox::information(this, tr("Modificar tarea."),
                                      tr("La tarea fue modificada con exito.\n"),
                                      QMessageBox::Ok);
          }
       }
    } else {

       if(Tarea_Is_The_Same(task, proyecto->getList_tarea()->at(pos.toInt()))){
```

```
            QMessageBox::information(this, tr("Modificar tarea."),
                                     tr("La tarea no fue modificada.\n"),
                                     QMessageBox::Ok);
        }
        else
        {
        if(clasificacion=="" ||  prioridad<=0  || nombre.size()<=0){
          if(clasificacion==""){
              QMessageBox m;
              m.setWindowTitle("Validar tarea.");
              m.setText("Debe elegir una clasificación.");
              m.exec();
          }
              if( prioridad<=0){
                  QMessageBox m;
                  m.setWindowTitle("Validar tarea.");
                  m.setText("La prioridad debe ser mayor que cero.");
                  m.exec();
              }
                  if(nombre.size()<=0){
                      QMessageBox m;
                      m.setWindowTitle("Validar tarea.");
                      m.setText("El campo nombre no puede estar vacio.");
                      m.exec();
                  }
        } else {
            proyecto->getList_tarea()->replace(pos.toInt(), task);
            LlenarTablaTarea(ui->tableWidget,proyecto->getList_tarea());
            Tarea_X_defecto();
            ui->lineEditvariablemodificar->setText("");
            ui->lineEditnombremodificar->setText("");
            ui->radioButtonanalisismodificar->setChecked(false);
            ui->radioButtondesarrollomodificar->setChecked(false);
            ui->radioButtondisennomodificar->setChecked(false);
            ui->radioButtonestudiomodificar->setChecked(false);
            ui->radioButtonpruebamodificar->setChecked(false);
            ui->spinBoxprioridadmodificar->setValue(0);
            QMessageBox::information(this, tr("Modificar tarea."),
                                     tr("La tarea fue modificada con éxito.\n"),
                                     QMessageBox::Ok);
  }
 }
}
   }
```

Anexo 6: Código para eliminar una tarea.

```
//Eliminar una tarea de la lista de tareas
void MainWindow::on_pushButton_3_clicked()
{
```

```
    QList<QTableWidgetItem*> filas_seleccionadas = ui->tableWidget->selectedItems();

   if(filas_seleccionadas.size()>0){
      for(int i=0; i < filas_seleccionadas.size(); i= i+4){
         if(proycreado){
            classe_Control->EliminarTarea(filas_seleccionadas.at(i)->text());
         }
         else
         {
           proyecto->EliminarTarea(filas_seleccionadas.at(i)->text());
         }
       }
      if(proycreado){
        LlenarTablaTarea(ui->tableWidget,classe_Control->getProy_Prueba()->getList_tarea());
      }
      else
      {
       LlenarTablaTarea(ui->tableWidget,proyecto->getList_tarea());
      }
      QMessageBox::information(this, tr("Eliminar tarea."),
                               tr("La(s) tarea(s) eliminada(s) con éxito.\n"),
                               QMessageBox::Ok);
   } else
   {
      QMessageBox::information(this, tr("Eliminar tarea."),
                               tr("Debe seleccionar alguna tarea a ser eliminada.\n"),
                               QMessageBox::Ok);
   }
}
```

Anexo 7: Código para validar el id de una tarea.

```
//Para validar el id de una tarea
bool MainWindow::ValidarIdTarea(QString id)
{
    bool idbueno = true;
    if(id.size()==0){
        QMessageBox m;
        m.setWindowTitle("Validar tarea.");
        m.setText("El campo variable no puede estar vacio.");
        m.exec();
        return false;
    }
    if(id.at(0)<'A' || id.at(0)>'Z'){
        QMessageBox m;
        m.setWindowTitle("Validar tarea.");
        m.setText("El identificador debe iniciar con mayúscula.");
        m.exec();
```

```
        idbueno = false;
    }
    if(id.at(0)>='0'&& id.at(0)<='9'){
        QMessageBox m;
        m.setWindowTitle("Validar tarea.");
        m.setText("El identificador no puede iniciar con número.");
        m.exec();
        idbueno = false;
    }
    for(int i=1; i<id.size();i++){
        if((id.at(i)<'A' || id.at(i)>'Z') && (id.at(i)<'a' || id.at(i)>'z') &&
                (id.at(i)<'0' || id.at(i)>'9')){
            QMessageBox m;
            m.setWindowTitle("Validar tarea.");
            m.setText("El identificador no puede contener caracteres especiales.");
            m.exec();
            idbueno = false;
            break;
        }
    }
    if(proycreado){
        for(int i=0;i<classe_Control->getProy_Prueba()->getList_tarea()->size(); i++){
            if(classe_Control->getProy_Prueba()->getList_tarea()->at(i)->getId_tarea().compare(id)==0){
                QMessageBox m;
                m.setWindowTitle("Validar tarea.");
                m.setText("El identificador ya está siendo utilizado.");
                m.exec();
                idbueno = false;
            }
        }
    } else {
        for(int i=0;i<proyecto->getList_tarea()->size(); i++){
            if(proyecto->getList_tarea()->at(i)->getId_tarea().compare(id)==0){
                QMessageBox m;
                m.setWindowTitle("Validar tarea.");
                m.setText("El identificador ya está siendo utilizado.");
                m.exec();
                idbueno = false;
            }
        }
    }
    return idbueno;
}
```

Anexo 8: Código para elegir el camino en el algoritmo de optimización.

```
//Elegir camino por cada hormiga
ArregloEnteros ControlClass::ElegirCamino(Feromona &matrix_feromona, ArregloEnteros
verticesVisitados, ArregloEnteros solucion, int posNodoActual)
```

```
{

   if(Recorrido_completo(verticesVisitados)){
      Depositar_Feromona(solucion,matrix_feromona);
     return solucion;
   }
   {
      verticesVisitados[posNodoActual] = 1;
      solucion.push_back(posNodoActual);
      QList<int> *amigos = Buscar_Amigos_dado_una_posicion(posNodoActual);
      double mayorProbabilidad = - 999.33435;
      int posNextNodo = 1000;
      for(int i=0; i < amigos->size(); i++){
         if(verticesVisitados[amigos->at(i)]==0){
            if(Probabilidad_de_elegir_el_siguiente_nodo(posNodoActual, amigos->at(i)) >
mayorProbabilidad){
               mayorProbabilidad = Probabilidad_de_elegir_el_siguiente_nodo(posNodoActual, amigos-
>at(i));
               posNextNodo = amigos->at(i);
            }
         }
      }
      if(posNextNodo == 1000){
        Depositar_Feromona(solucion,matrix_feromona);
          return solucion;
      }
      return ElegirCamino(matrix_feromona, verticesVisitados, solucion,posNextNodo);
   }
}
```

Anexo 9: Código para depositar feromona.

```
void ControlClass::Depositar_Feromona(ArregloEnteros solucion, Feromona &matrixFeromona)
{
    int aportes = 0;
   for(int i=0;i < solucion.size()-1; i++){
      aportes += getMatrix_peso_aristas().at(solucion.at(i))[solucion.at(i+1)];

   }

   for(int i=0; i < solucion.size()-1; i++){

      matrixFeromona.at(solucion.at(i))[solucion.at(i+1)] += (double)(getQu()/aportes);

       matrixFeromona.at(solucion.at(i+1))[solucion.at(i)] =
matrixFeromona.at(solucion.at(i))[solucion.at(i+1)];

   }
```

```
}
```

Anexo 10: Código para actualizar feromona.

```
void ControlClass::Actualizar_Feromona(Feromona &matrixFeromona, int numSoluciones)
{
    double aportes = Suma_de_los_aportes(numSoluciones);
    for(int i=0; i < matrixFeromona.size(); i++){
        for(int j= i; j< matrixFeromona.size(); j++){
            if(i==j)
                matrixFeromona[i][j] = 0;
            matrixFeromona[i][j] = ((1-getRo())*matrixFeromona[i][j])+ getQu()/aportes;
            matrixFeromona[j][i] = matrixFeromona[i][j];
        }
    }
}

double ControlClass::Probabilidad_de_elegir_el_siguiente_nodo(int nodo1, int nodo2)
{
    int alfa = getAlfa(), beta = getBeta();
    double feromona = Feromona_en_un_camino(nodo1, nodo2),
          visibilidad = (double)Visibilidad_en_un_camino(nodo1,nodo2);
    double numerador = pow(feromona,alfa) * pow(visibilidad,beta);
    double denominador = Sumatoria_de_los_posibles_caminos(nodo1);

    return numerador/denominador;
}
```

1