Universidad Zaragoza

TRABAJO FIN DE GRADO

DISEÑO DE SOFTWARE PARA ORGANIZACIÓN Y GESTIÓN ADAPTATIVA DE PROYECTOS

SOFTWARE DESIGN FOR ADAPTIVE PROJECT ORGANIZATION AND MANAGEMENT

Autor

Javier Martínez Lahoz

Tutor

Tomás Cortés Arcos

Escuela Universitaria de La Almunia

2019

Escuela Universitaria
Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA

DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

MEMORIA

DISEÑO DE SOFTWARE PARA ORGANIZACIÓN Y GESTIÓN ADAPTATIVA DE PROYECTOS

SOFTWARE DESIGN FOR ADAPTIVE PROJECT ORGANIZATION AND MANAGEMENT

425.17.42

Autor: Javier Martínez Lahoz

Tutor: Tomás Cortés Arcos

Fecha: Septiembre 2019

Índice de contenido

Índice de figuras

1. Resumen

Este trabajo está orientado a diseñar un software informático que permita gestionar de forma automática los recursos humanos en un proyecto. Se considera que esta gestión consiste en la asignación de trabajadores a las tareas de acuerdo a sus características y al estado del proyecto.

Para poder automatizar un proceso, el primer paso consiste en desarrollar un modelo teórico de los elementos implicados. El modelado de los trabajadores y las tareas requiere, a su vez, del estudio de un modelo teórico. Se ha escogido el modelo de la gestión por competencias, ya que permite relacionar de forma efectiva los atributos de los trabajadores y las tareas. Este método permite asignar valores numéricos a las aptitudes de los trabajadores y a las necesidades de las tareas, por lo que se facilita su implantación en un sistema informático.

Las competencias se emplean para determinar la eficacia de un trabajador en una tarea concreta. Sin embargo, son los conocimientos y destrezas del trabajador las que definen qué tareas puede realizar. Por lo tanto, se deben tener en cuenta ambas características, tanto del trabajador como de la tarea, para realizar el reparto de responsabilidades.

Para demostrar la viabilidad del software diseñado y poner a prueba los algoritmos de gestión y asignación, se decide llevar a cabo una demostración práctica con los modelos. Este programa de prueba cuenta con las funciones necesarias para realizar la gestión y asignación de recursos en un proyecto. Además, esta aplicación sirve para adquirir una mayor información acerca de las necesidades de una implementación real a gran escala.

En este programa de prueba se pueden introducir tareas, su duración correspondiente y las dependencias entre ellas para su posterior procesado. El primer cómputo consiste en realizar la organización temporal de las tareas, incluyendo el cálculo de la ruta crítica y las holguras temporales. Si también se introducen trabajadores al programa, este los asigna a cada tarea de acuerdo a sus conocimientos, destrezas y competencias. La aplicación comienza por la asignación de las tareas de la ruta crítica, y asigna siempre al trabajador disponible que se considere más adecuado.

Tras verificar el correcto funcionamiento de la aplicación de demostración, se procede a realizar el diseño de un programa que permita la implementación a gran escala en una empresa real. Para ello resulta necesario estudiar factores que no afectan directamente a la aplicación de demostración. Entre ellos destacan el diseño de la arquitectura del sistema, los diferentes niveles de acceso, la estructura de la base de datos o la mayor cantidad de información que es necesario procesar.

Se considera que la implementación de un software de estas características en una empresa real es posible, siempre que se realicen las modificaciones necesarias. Aun así, sería necesario llevar a cabo un periodo de pruebas y adaptación para asegurar el correcto funcionamiento. Esto sería más importante aún en el caso de integrar el software en un sistema ERP ya implantado. Su principal ventaja sería facilitar la gestión de proyectos y el trabajo del departamento de recursos humanos, resultando útil para empresas de cualquier tamaño.

1.1 Palabras clave

Recursos humanos, proyectos, gestión, informática, python

2. Abstract

The objective of this project is to design a project organization and management software. This management consist of the correct distribution of tasks among the workers, according to their characteristics and the project status.

In order to automate a process, the involved elements must be modeled first. Therefore, a theoretical model should be studied, ensuring that an appropriate model is developed. The competency management system has been chosen for this project, since it relates the attributes of the workers with the requirements of the tasks. This method allows to assign numeric values to the characteristics of workers and tasks; simplifying the implementation in an information system.

The competencies of the workers are used to determine its efficiency in the tasks. However, the skills and knowledge of the workers define what tasks they can perform, and which they cannot. Consequently, all these characteristics, both of the workers and of the tasks, must be taken into account in the distribution of responsibilities.

A demonstration software is developed to test the management algorithms and ensure the viability of the proposed software. This test platform includes the necessary functions to manage and distribute human resources in one project. This program also allows to acquire information about the requirements of a larger scale implementation.

The demonstration software allows the user to insert tasks, with their corresponding duration and dependencies. The software performs the temporary distribution of these tasks, including the critical path and temporary clearances computation. If workers are also included in the software, it will distribute them among the tasks according to their knowledge, skills and competencies. The first tasks assigned will be those of the critical path, and the assigned worker will be the most suitable.

Once the demonstration software works as intended, and all the information has been collected, software can be designed on a larger scale. This software is intended to be implemented in a company. Therefore many factors, such as the hardware architecture, access permissions, data base structure or information flow, must be taken into account in this new design.

The implementation in a real company is considered as a feasible possibility. This software would help to improve the performance of the human resource department for any size of company.

3. Objetivo

La meta de este trabajo es diseñar un software informático que permita gestionar de forma automática la organización de un proyecto. Para poder llevar a cabo este diseño de forma satisfactoria es necesario cumplir una serie de objetivos intermedios.

El primer objetivo se basa en analizar los elementos implicados en la asignación: los trabajadores y las tareas del proyecto. Con estos análisis se busca relacionar las características de ambos elementos entre sí con la finalidad de determinar el nivel de desempeño de un trabajador concreto en una tarea determinada.

El segundo objetivo consiste en modelar los trabajadores y las tareas. De acuerdo a estos modelos se diseñará una aplicación informática de demostración. En la misma también se llevará a cabo la implementación de las relaciones entre los elementos mediante funciones o algoritmos. Se desarrollará esta aplicación hasta lograr un programa funcional para organizar y gestionar un proyecto. Se buscará que la aplicación sea capaz de organizar temporalmente las tareas, y de distribuirlas entre los trabajadores. Y, además, se pretende que la aplicación cuente con una interfaz gráfica que facilite el control y la visualización de información por parte del usuario.

El objetivo final de este trabajo consiste en el diseño una aplicación, basada en la implementada previamente, que permita gestionar la totalidad de los proyectos y trabajadores de una organización.

4. Introducción

4.1 Motivación

El desarrollo de proyectos es, en su mayor parte, una actividad de equipo. Una de las tareas más influyentes en la viabilidad de un proyecto es la asignación los trabajadores a las diferentes tareas. “La productividad y elevada calidad aparecen cuando los miembros del equipo contribuyen de forma efectiva y permanecen motivados, además el equipo en general funciona de forma eficiente y sin problemas” (Pankaj Jalote, 2002, Software Project Management in Practice). La gestión de recursos en un proyecto es una tarea fundamental para lograr que este se desarrolle de forma adecuada. Una gestión inadecuada de los recursos, ya sean humanos o materiales, puede hacer fracasar el proyecto.

Incluso cuando la organización inicial es adecuada, los plazos y los requerimientos pueden variar durante el transcurso del proyecto. Cualquier tipo de alteración respecto al planteamiento original, como el retraso de una tarea, la baja de un trabajador o la aparición de nuevos objetivos, pueden desencadenar graves problemas en el proyecto.

Para minimizar estos contratiempos, y evitar que el proyecto pueda fracasar, se debería implementar un método de organización flexible. Esto permitiría realizar variaciones en el reparto de tareas y recursos, afrontando cualquier nueva situación a la mayor brevedad. También puede resultar interesante aplicar modificaciones en la asignación de las tareas a los trabajadores en función de su desempeño en tareas anteriores. Esto no se realizará para adaptarse a nuevas circunstancias, sino con el objetivo de maximizar la eficiencia del trabajo y la motivación de los empleados en todo momento.

En proyectos con pocos trabajadores, o muy diferenciados entre sí, puede resultar sencillo asignar el trabajador que realizará cada tarea. Sin embargo, cuando los trabajadores son polivalentes y el número de tareas es elevado puede resultar más complejo realizar esta asignación. Una asignación eficiente será aquella que logre que todos los trabajadores mantengan una carga de trabajo similar, evite los tiempos desocupados y asegure la viabilidad general de los proyectos.

Para lograr una modificación de la planificación del proyecto durante su curso resulta necesario mantener un control exhaustivo del estado del proyecto. Esto requiere de una gran cantidad de recursos invertidos, especialmente si se controlan varios proyectos de forma simultánea. La gran cantidad de información acerca de los trabajadores y las tareas puede complicar en gran medida la gestión manual de los proyectos.

La solución planteada en este trabajo consiste en automatizar esta gestión de los proyectos. El uso de una herramienta informática permitiría trabajar con una gran cantidad de información, analizando los proyectos de forma continua. Mediante esta aproximación se espera conseguir un método de organización que se vaya adaptando a las diferentes situaciones de forma automática, antes de que puedan afectar negativamente a la viabilidad del proyecto.

4.2 Método de trabajo

En primer lugar es necesario analizar la base teórica sobre la que se diseñará el sistema automatizado de gestión. El punto más importante de la misma se considera la distribución de las tareas entre los trabajadores.

Por lo tanto, se va a realizar un estudio sobre la psicología del trabajo, concretamente sobre los métodos de análisis de trabajadores y tareas, orientados a la asignación de puestos o responsabilidades.

El análisis de los trabajadores debe servir para estimar el rendimiento de cada trabajador en una determinada tarea. Para ello, se comenzará por determinar los factores que capacitan, o no, a un trabajador para llevar a cabo una determinada tarea. Tras esto, se estudiarán los factores implicados en el nivel de desempeño. Dentro de estos factores se prestará una especial atención a los métodos de asignación por competencias.

Como se verá más adelante, los métodos de gestión por competencias se pueden adaptar para realizar los cálculos y las asignaciones de forma automática. Por lo tanto, será necesario efectuar un análisis previo sobre la gestión por competencias y los métodos de evaluación.

Cualquier trabajador puede destacar en el desempeño de unas tareas, pero será incapaz de llevar a buen término otras. Esto se debe a los diferentes requisitos de las diferentes tareas. También resultará necesario realizar un análisis acerca de las características de las tareas. Este análisis se realizará de forma análoga al de los trabajadores. Esto facilita los métodos empleados para relacionar ambos. En primer lugar se estudiarán los aspectos que limitan los trabajadores que pueden ejecutar una tarea. Y, en segundo lugar, se elaborará un análisis competencial para determinar qué perfil de trabajador se adapta mejor a una determinada tarea.

Al combinar estos dos análisis se busca desarrollar un método que sirva para diferenciar, dentro de un grupo cerrado de trabajadores, a aquellos aptos para realizar una tarea concreta. De entre aquellos capaces, se escogerá al trabajador que, de acuerdo a las competencias estudiadas, tenga mejor rendimiento estimado para la tarea.

Tras el análisis teórico de los elementos involucrados en la asignación, se pasa a realizar un estudio práctico acerca de la posibilidad de esta implementación. Para ello, se opta por desarrollar una aplicación informática. Esta aplicación servirá para poner en práctica los estudios teóricos realizados anteriormente y obtener una mayor información de cara al diseño software final. La aplicación desarrollada contará con una interfaz gráfica para facilitar su uso por parte de cualquier usuario e implementará los algoritmos necesarios para demostrar la viabilidad técnica del trabajo.

En la parte final se estudiará la implementación a gran escala en una organización. Para ello se habrán de tenerse en cuenta todos los aspectos comunes con la versión a pequeña escala, pero también elementos propios debidos a las nuevas condiciones de trabajo.

4.3 Antecedentes

En el mercado existen una gran cantidad de herramientas informáticas orientadas a la gestión en organizaciones. Sin embargo, ninguno de ellos parametriza los trabajadores ni las tareas, por lo que la asignación de puestos debe realizarse de forma manual.

Esto no suele ser un problema, ya que la mayoría de las empresas utilizan puestos fijos, por lo que la asignación es una decisión a largo plazo que se realiza una vez.

En el mercado existen dos clases de herramientas relacionadas con el objetivo de este trabajo: los sistemas de planificación de recursos empresariales y los sistemas de administración de proyectos.

4.3.1 Sistemas de planificación de recursos empresariales

Los sistemas de planificación de recursos empresariales, o *ERP* (*Enterprise Resource Planning*), son sistemas informáticos empleados para gestionar los procesos y recursos de una organización. Estos sistemas suelen estar compuestos por varios módulos interrelacionados entre sí, donde cada uno tiene una funcionalidad distinta. Entre los módulos más implantados se encuentran aquellos relacionados con producción, ventas, logística, contabilidad, etc. Estos sistemas tienen como objetivos principales la optimización de procesos y la mejora del acceso a la información. Los módulos de recursos humanos generalmente están orientados a costes, gestión de turnos y procesos de incorporación de nuevos trabajadores.

Existen una gran cantidad de aplicaciones de tipo *ERP* que permiten gestionar de forma efectiva las empresas. Entre los sistemas más importantes en el mercado actual podemos destacar *SAP ERP* y *Odoo*.

SAP ERP

SAP es una de las aplicaciones de planificación de recursos con más tiempo en el mercado, fue desarrollado por la compañía alemana *SAP SE* en 1992. Se trata de una herramienta formada por varios bloques con una base de datos común. Cada uno de estos bloques representa un proceso de un departamento de la organización. Entre estos bloques encontramos aquellos relacionados con las finanzas, la logística, los recursos humanos, el control de producción…

Se trata de una solución muy completa y profesional, que gestiona y distribuye de forma autónoma toda la información de la empresa a los departamentos adecuados. El principal inconveniente de esta herramienta es su coste, ya que es necesario recurrir a sus profesionales para realizar la implantación y cualquier modificación posterior. Su principal ventaja es precisamente la profesionalidad del equipo con el que cuenta.

Odoo

Odoo es una herramienta de gestión integral de código abierto. Esto ha facilitado que su comunidad desarrolle una gran cantidad de módulos para todas las necesidades empresariales.También cuenta con una versión con licencia, que es instalada por un equipo profesional, de forma similar a *SAP ERP*.

Entre los módulos disponibles está la gestión de compraventa, clientes, proyectos, almacenes, manufactura, contabilidad, recursos humanos, inventario, márketing… Todos estos módulos se pueden activar o desactivar dentro de una misma plataforma.

Una de sus principales ventajas es que, al tratarse de una aplicación de código abierto, se puede personalizar y modificar como la empresa desee. Por lo que otorga una mayor flexibilidad frente a otros programas ERP propietarios como SAP.

Su principal inconveniente si se emplea la versión libre, es que la instalación debe ser realizada por la propia empresa, por lo que no existen garantías de éxito.

4.3.2 Sistemas de administración de proyectos

Los sistemas de administración de proyectos, o *PMS* (*Project Management Software*), se emplean para la planificación y gestión de proyectos. En la actualidad existen numerosas soluciones *PMS* y se usan en una amplia variedad de organizaciones. Esto se debe a su

utilidad para planificar y estimar plazos, controlar costes y presupuesto, asignar recursos, comunicar información, gestionar documentación, etc.

Un *PMS* tiene dos objetivos fundamentales: la planificación y la gestión de información. La planificación, que puede llevarse a cabo mediante distintas metodologías sirve para conocer en todo momento el estado y la previsión de las diferentes tareas del proyecto. Para realizar esta previsión se analizan las relaciones entre tareas, necesidades de recursos y cálculos de duración mediante modelos probabilísticos.

Microsoft Project

Microsoft Project, *MSP,* es la herramienta desarrollada y comercializada por Microsoft para la administración de proyectos. Se emplea para ayudar a los administradores de proyectos a desarrollar planes, asignar recursos a tareas, generar seguimiento y administrar el presupuesto. La aplicación permite organizar temporalmente los proyectos; para ello calcula las rutas y cadenas críticas. También secuencia las tareas de acuerdo a los requisitos de recursos del proyecto y su disponibilidad.

Esta aplicación esta especialmente orientada a la generación de información en relación al seguimiento de los proyectos. Se pueden obtener rápidamente informes y diagramas, como el GANNT, para conocer el estado del proyecto y tomar decisiones en consecuencia.

Uno de los principales puntos negativos de *MSP* es que, al tratarse de un software con licencia, tiene un coste económico mensual, que aumenta por cada usuario que se incorpore al sistema.

GANNT Project

GANNT Project es una herramienta *PMS* de código abierto. Se trata de una aplicación sencilla, basada en el principio *KISS* (*Keep it stupid simple*); este defiende que los sistemas funcionan mejor cuanto más sencillos son. Por lo tanto, toda complejidad innecesaria debe ser evitada.

Esta herramienta permite crear diagramas de GANNT para controlar la distribución temporal de las tareas y gestionar el uso de recursos. Al establecer jerarquías y dependencias entre las tareas, la aplicación es capaz de generar diagramas PERT para el análisis de las rutas críticas del proyecto.

Entre sus mayores inconvenientes están la incapacidad de trabajar en el marco temporal de horas, y la falta de gestión de mensajes o documentación relacionada con las tareas.

4.4 Competencias

4.4.1 Definición de competencia

Se pueden definir una competencia como una característica subyacente de un individuo que está causalmente relacionada con una efectividad o desempeño bueno o excelente en un trabajo o situación concreta. Si se analiza esta definición, se puede comprender mejor su significado. Para ello conviene examinar los conceptos de “característica subyacente” y “relación de causalidad”.

Una característica subyacente es aquella que forma parte de la personalidad de un individuo y que puede predecir su comportamiento en diversas situaciones. Una relación de

causalidad significa, en este caso, que la competencia anticipa el comportamiento y el desempeño del individuo en una tarea concreta. Todas las personas cuentan con un conjunto competencias que determinan cuál será su comportamiento frente a una determinada actividad.

4.4.2 Clasificación de las competencias

Las competencias se pueden clasifican en cinco agrupaciones: conocimientos, habilidades, rasgos de personalidad, características y motivación.

Conocimiento

El conocimiento es el conjunto de informaciones que una persona posee sobre áreas específicas. Se consideran tanto los conocimientos técnicos como los referidos a las relaciones interpersonales. Algunos ejemplos son el conocimiento del mercado, de informática, de la anatomía humana, el manejo de situaciones conflictivas, etc.

En muchas ocasiones se consideran únicamente los conocimientos memorísticos o teóricos de un trabajador. Sin embargo, resulta más importante que un individuo sepa cómo actuar frente a un determinado problema, buscando información en las fuentes adecuadas y empleando aquella de mayor utilidad. Es decir, prima saber aprovechar el conocimiento, antes que disponer de una gran cantidad de conocimiento, pero no saber emplearlo.

También conviene tener en cuenta que la mayoría de las evaluaciones de conocimiento sirven para medir la información que posee el trabajador, no si es capaz de actuar de acuerdo a dicha información. Por ejemplo, la habilidad de determinar el mejor argumento es muy diferente a la habilidad para solucionar una situación conflictiva mediante la persuasión. Es decir, el conocimiento de un trabajador representa lo que es capaz de hacer, y no lo que hará en una determinada situación.

Habilidades

Las habilidades, también denominadas en ocasiones aptitudes o destrezas, representan la capacidad de un trabajador para desempeñar una determinada tarea o actividad, ya sea física o mental. Supone aplicar los conocimientos que posee para resolver los problemas que le plantea su trabajo. Se trata de un tipo de competencias que se debe evaluar de forma práctica.

El razonamiento numérico, la realización de presentaciones, la soldadura de metales o la capacidad de programación son algunos ejemplos de habilidades.

Rasgos de personalidad

Existen una gran cantidad de rasgos de personalidad que afectan al modo en que un trabajador desempeña una determinada tarea. Estos rasgos generan una predisposición general a comportarse o reaccionar de un modo determinado.

La tenacidad, el autocontrol, la resistencia al estrés o la orientación al trabajo en equipo son algunos rasgos que puedan afectar en gran medida a la forma en la que un trabajador lleva a cabo una determinada tarea en una situación concreta.

En esta categoría también se suele incluir el concepto de uno mismo. Este engloba rasgos como la confianza en sí mismo o la seguridad en poder desempeñar bien las acciones.

Esta seguridad o confianza influye en gran medida en las decisiones que un trabajador tomará en una determinada situación. Y ello resulta especialmente importante cuando el trabajador se encuentra en un puesto de responsabilidad o con trabajadores a su cargo.

Características

En esta agrupación se incluyen las características físicas y respuestas estables a situaciones o información. Se trata de los rasgos propios de cada trabajador que afectan a cómo desempeñará una determinada actividad. El tiempo de reacción o la buena vista son características que pueden afectar a cómo un trabajador lleva a cabo una determinada tarea, aunque no se trata de conocimientos, destrezas ni rasgos de personalidad.

Estos rasgos no se pueden aprender, pero sí pueden variar a lo largo del tiempo debido a una gran cantidad de motivos internos o externos.

Motivación

Se puede definir la motivación como los intereses que una persona desea o considera. Estos intereses afectan al comportamiento de los individuos, fomentando unas determinadas acciones y evitando otras. La motivación esta basada en necesidades o formas de pensar.

La motivación es una característica difícil de promover activamente. Sin embargo, puede alterarse o sufrir altibajos a gran velocidad debido a una amplia variedad de factores, tanto internos (de la organización) como externos.

4.4.3 Evaluación de competencias

A la hora de evaluar las competencias hay que tener en cuenta que no todas las competencias se evalúan igual. El modelo del Iceberg divide de forma gráfica las competencias en dos grandes grupos en función de la facilidad de evaluación.

Fig. 1: Modelo del Iceberg (blog UPCplus by CERpIE-UPC)

De acuerdo al modelo del iceberg, las destrezas y lo conocimientos se encuentran en la superficie, por lo que resultan las competencias más sencillas de desarrollar y evaluar. Sin embargo, los rasgos de personalidad, el concepto de uno mismo y las actitudes se encuentran debajo de la superficie, siendo más difíciles de detectar, evaluar y, especialmente, desarrollar.

Las habilidades y los conocimientos pueden ser evaluados mediante pruebas teóricas o prácticas. También pueden ser desarrolladas fácilmente mediante cursos de formación, ya sean en la propia empresa o en una organización externa. Este es el papel de los contratos de beca o de prácticas remuneradas: desarrollar las habilidades y los conocimientos de los posibles nuevos trabajadores, que tienen menos experiencia, de cara a sus futuros puestos.

Las actitudes y algunos rasgos de personalidad se pueden identificar observando el comportamiento de los trabajadores en su trabajo diario. Aunque, para evaluar adecuadamente los rasgos más profundos de la personalidad y la motivación puede ser necesario realizar un estudio psicológico del trabajador. Estas características personales suelen ser muy difíciles de desarrollar o modificar, por lo que suele ser recomendable escoger a los trabajadores para un puesto determinados en función de ellas.

Sin embargo, también resulta complicado lograr un análisis fiable en estos ámbitos antes de realizar la contratación de un nuevo trabajador. Esta también es una de las ventajas de la contratación y la promoción interna: tener una mayor información acerca del perfil psicológico de los aspirantes al puesto.

4.4.4 Gestión por competencias

La gestión por competencias es una herramienta de gestión de los recursos humanos de una organización. Este modelo de gestión intenta alinear los intereses de los profesionales de una empresa con los intereses de la propia organización.

Un modelo de gestión por competencias necesita que la organización defina su política de empresa. Dentro de esta política se debe incluir la definición y nivelación de las competencias necesarias para que se logren los objetivos propuestos en cada puesto de trabajo. Estas competencias pueden diferenciarse en dos grupos: competencias genéricas y competencias específicas.

Las competencias genéricas son aquellas alineadas con los objetivos de la organización y, por lo tanto, aquellas que todos los trabajadores deberían tener. En esta categoría podemos incluir el compromiso con la organización, la comprensión social, la orientación al cliente (interno y externo).

Las competencias específicas son aquellas que dependen del puesto de trabajo. Su objetivo es mejorar el desempeño de los trabajadores en sus tareas cotidianas. Por lo tanto, varían en función del puesto de trabajo que se analice. El liderazgo, la iniciativa, la comunicación, la cooperación o la flexibilidad son algunos ejemplos de competencias específicas.

En los puestos complejos, las competencias resultan de mayor importancia que las habilidades relacionadas con la tarea. Lo que suele distinguir a los trabajadores con mayor desempeño son la motivación y las habilidades interpersonales y políticas. La mejor decisión en estos casos, aunque también es útil en el resto, es seleccionar en base a buenas competencias y motivación; el conocimiento y las destrezas requeridas para las tareas se pueden enseñar posteriormente mediante cursos de formación.

5. Criterios de asignación

En este apartado se realiza el análisis de los trabajadores y las tareas necesarios para poder llevar a cabo el desarrollo software posterior. Para ello se empleará el modelo de gestión por competencias orientado a una futura implementación software.

La mayoría de los proyectos de cualquier organización cuentan con varias tareas e involucran a más de un trabajador. Si el volumen de trabajo es muy elevado, se suelen emplear grupos de trabajo, a los que se les asigna una rama del proyecto. Estas ramas se dividen a su vez en tareas atómicas para poder repartirlas entre los miembros del equipo.

Si se dividen las ramas en sus tareas atómicas de forma adecuada, se puede asumir que cada una tiene un único responsable. En algunos casos se puede considerar que dos trabajadores realizan una tarea de forma completamente cooperativa. Sin embargo, de cara a la gestión del proyecto, esto se puede simplificar como dos trabajadores realizando dos tareas idénticas con la restricción de que comiencen a la vez.

Una vez definidas las tareas, es necesario distribuir los trabajadores entre las actividades de forma óptima. Esta asignación es una tarea compleja, así como un elemento fundamental que afecta en gran medida y que puede determinar el éxito o el fracaso del proyecto. Una asignación realizada puede ser modificada en cualquier momento, pero esto puede repercutir negativamente en el desarrollo del proyecto (objetivos no claros, cambio de responsable de actividades…).

Para poder realizar una asignación adecuada, el responsable del proyecto debe tener en cuenta una gran cantidad de factores de cada trabajador. Entre estos factores están la experiencia, destrezas, conocimientos, aspiraciones, trayectoria profesional, competencias… Por lo general, no se emplea un modelo matemático, sino que el responsable del proyecto toma decisiones basándose en su experiencia y conocimiento de los trabajadores y de las tareas. Si el encargado de la asignación no posee el suficiente conocimiento acerca de los trabajadores y las tareas, esta asignación puede no ser la adecuada. Esto provocará retrasos, y podrá llegar a afectar a la viabilidad del proyecto.

El objetivo de este trabajo consiste en automatizar este proceso de asignación. Para poder automatizar una operación es necesario contar con información objetiva y modelos teóricos acerca de los elementos implicados. Para modelar de forma correcta los elementos se deben analizar de forma individual, pero también las relaciones entre ellos.

5.1 Análisis de trabajadores

En el análisis de los trabajadores se deben estudiar aquellas características que afectan a su desempeño profesional, es decir, sus competencias. Existen diferentes enfoques para analizar los factores humanos implicados. Los métodos orientados al trabajador analizan las competencias necesarias para llevar a cabo una tarea de forma satisfactoria.

Entre estos sistemas de análisis encontramos el *F-JAS*, sistema de análisis con enfoque al trabajador desarrollado por Edwin A. Fleishman. En este método se estudian tres áreas mediante un sistema de 73 escalas. En el área cognitiva se evalúan las habilidades verbales, la generación de ideas, la memoria, la atención, etc. En el área psicomotriz se evalúan las habilidades de manipulación fina, el movimiento de control, el tiempo de reacción, la velocidad, etc. Y en el área física se evalúan las habilidades de fuerza física, resistencia, flexibilidad, agudeza visual y auditiva, etc.

Al relacionar este método con las competencias del apartado 4.4, se puede observar que se trata en su mayor parte de competencias genéricas, es decir, características del trabajador. Este método no analiza los conocimientos o las destrezas de un trabajador que tienen relación directa con el desempeño de una tarea específica.

Otro método que se emplea de forma habitual es el método *KSAO*. Este método también clasifica los atributos humanos en áreas, pero lo hace de forma diferente al sistema *F-JAS*. Las escalas del método *F-JAS* se basan, en su mayor parte, en características; sin embargo, el método *KSAO* clasifica las competencias en cuatro categorías: conocimientos (*Knowledge*), destrezas (*Skills*), habilidades (*Abilities*) y otras características (*Other*).

Este método permite disponer de una mayor información de los trabajadores, incluyendo aspectos psicológicos, incluidos en la categoría de otras características.

Fig. 2: Clasificación de los atributos en el modelo *KSAO*

Este método es muy similar a los métodos de clasificación de competencias expuestos en el apartado 4.4. La diferencia es que el método *KSAO* denomina competencias (*abilities*) únicamente a los rasgos de personalidad que afectan directamente al desempeño en una tarea (liderazgo, flexibilidad, etc).

Para evitar confusiones se va a emplear esta distinción a partir de este punto del trabajo; denominando “competencias” a aquellas relacionadas con la personalidad y las habilidades no técnicas.

Se ha decidido estudiar el método *KSAO* y acompañarlo con los modelos de gestión por competencias para actuar como base para el modelo del trabajador que, posteriormente, se implementará en el software.

5.1.1 Conocimientos

De acuerdo al método *KSAO*, el conocimiento es el conjunto de información que un trabajador posee sobre un determinado campo. Esta información puede haber sido obtenida mediante educación formal, práctica, o cualquier otro método. Esto encaja perfectamente con la definición de conocimiento de la gestión por competencias del apartado 4.4.2

Se considera que los conocimientos pueden ser valorados de forma absoluta, es decir, no puede tenerse parte de un conocimiento. Se puede suponer que los conocimientos no se

pierden debido a la falta de práctica, pero si que se incrementan mediante cursos formativos o experiencia.

Los conocimientos cuentan con niveles, en función de la cantidad y calidad de información que posea el trabajador. Por ejemplo, de forma análoga al sistema educativo se pueden clasificar los niveles de conocimiento en nivel básico, de grado medio, de grado superior, de grado universitario, de máster o de doctorado.

Modelar los conocimientos por niveles puede aumentar la complejidad de los modelos y los algoritmos posteriores. Sin embargo, de acuerdo al alcance del trabajo, y con el objetivo de facilitar el posterior modelado de los trabajadores y las tareas, se van a tratar los conocimientos de distintos niveles como conocimientos distintos; y no como un único conocimiento en distintos grados.

El conjunto de conocimientos de un trabajador determina los puestos de trabajo que puede ocupar de forma efectiva. Por lo tanto, esta variableresulta un elemento fundamental que debe ser incluido en el modelo informático.

Sin embargo, el conocimiento no siempre implica ser capaz de realizar una tarea. Puede darse el caso de que un trabajador posea el conocimiento necesario para llevar a término una determinada tarea pero, por motivos físicos, no la pueda llevar a cabo. Esto se puede deber a las destrezas o a las características del trabajador. Un ejemplo claro de esta situación son los entrenadores deportivos.

5.1.2 Destrezas

La destreza se define como la habilidad y experiencia para la realización de una actividad determinada. En el caso del método de evaluación *KSAO*, se consideran destrezas a las habilidades relacionadas con la realización del trabajo.

La relación directa con unas tareas determinadas es un punto en común entre los conocimientos y las destrezas. Sin embargo, si un trabajador tiene una destreza, implica necesariamente que es capaz de ejecutar la actividad relacionada. Si, por cualquier motivo, un trabajador ya no es capaz de realizar dicha tarea, quiere decir que ya no tiene la destreza necesaria.

Las destrezas se pueden obtener y mejorar mediante la práctica y la experiencia, pero se pierden progresivamente cuando no se practican. Por esto podemos considerar que las destrezas, al igual que los conocimientos, pueden poseerse en distintos grados, en función de la práctica y la habilidad innata.

Para facilitar el modelado posterior de los trabajadores y las primeras pruebas, las destrezas se van a computar de forma similar a los conocimientos, considerando únicamente si un trabajador posee, o no, una determinada destreza. Distintos grados de habilidad se modelarán como destrezas independientes. Al igual que en el caso de los conocimientos, en un futuro se podrán estudiar otros métodos de modelado.

5.1.3 Competencias

Según el método KSAO, las competencias son el conjunto de capacidades estables en el tiempo. A diferencia de los conocimientos o las destrezas, las competencias no están relacionadas con una tarea concreta, sino que son genéricas y se pueden emplear en una gran

variedad de situaciones distintas. Por lo tanto, las competencias no deberían determinar qué tareas puede realizar un trabajador, pero sí que afectarán a su nivel de desempeño.

Existen una gran diversidad de competencias genéricas que se pueden evaluar en los trabajadores: adaptabilidad, capacidad crítica, flexibilidad, capacidad de comunicación, liderazgo, iniciativa… La evaluación de las competencias en un trabajador resulta más complicada que la evaluación de conocimientos o destrezas. Ya que, como se ha visto anteriormente, no se puede realizar mediante una prueba teórica, práctica, observación o un análisis psicológico.

Todos los trabajadores poseen un determinado nivel para cada competencia. Resulta necesario emplear un método de evaluación para obtener una valoración objetiva de cada una de las competencias evaluadas. Un método habitual consiste en realizar un análisis del trabajador en su ambiente de trabajo o, en determinados casos, un análisis psicológico. Esto suele ser necesario para las competencias más ligadas a los rasgos de la personalidad o las motivaciones personales.

La cantidad de competencias evaluadas en cada trabajador dependerá de los recursos que la organización responsable esté dispuesta a invertir. Evaluar pocas competencias puede no generar la suficiente información, y por lo tanto, conducir a decisiones equivocadas; un exceso de competencias evaluadas puede incrementar el coste sin proporcionar información útil adicional.

Métodos de evaluación

Los métodos de evaluación implantados en las empresas suelen basarse en formularios, que deben ser desarrollados por profesionales especializados para que su resultado sea fiable. En el proceso, un supervisor evalúa el comportamiento de sus trabajadores durante el transcurso de sus actividades. En estos formularios se responde a preguntas mediante unas respuestas predefinidas o una escala numérica.

En la tabla de la figura 3 se muestra un ejemplo extraído de un formulario real empleado para la evaluación del liderazgo en trabajadores. Permite obtener un resultado numérico para comparar fácilmente la valoración de una competencia entre distintos trabajadores. Se puede observar cómo cada pregunta del formulario evalúa un comportamiento concreto. Las respuestas a esta pregunta se realizan de forma numérica, aunque se encuentran relacionadas con una expresión lingüística (Insuficiente, Normal, Satisfactorio, Bueno o Excelente).

Capacidad de liderazgo	Ins	Nm	Sat	Bue	Exc
Empuja a la consecución de buenos resultados	1	2	3	4	5
Transmite sentido de urgencia cuando es necesario	1	2	3	4	5
Dirige y trabaja para resolver problemas	1	2	3	4	5
Crea ambiente para que se disfrute trabajando	1	2	3	4	5
Consigue facilmente la confianza y el respeto de los demás	1	2	3	4	5

Fig. 3: Formulario de evaluación de liderazgo

El agente evaluador puede afectar a la evaluación de la competencia, ya que la valoración de cada respuesta está sujeta a interpretación. Lo ideal es realizar más de una evaluación, a cargo de distintos evaluadores, para intentar conseguir un resultado más representativo de la realidad. Sin embargo, los recursos adicionales empleados para la evaluación no necesariamente generan más información.

Las competencias de cada trabajador pueden variar según este desarrolla distintas tareas. Por lo tanto, resulta indispensable realizar evaluaciones de desempeño de forma periódica. Puede ser interesante llevar a cabo evaluaciones tras cada proyecto. Esto permitiría que el desempeño de los trabajadores en cada proyecto fuera un factor influyente en el reparto de tareas del siguiente.

5.1.4 Otras características

Existen determinadas características personales que afectan al desempeño profesional de un trabajador, pero que no se pueden incluir en ninguna de las categorías anteriores. Estos atributos suelen estar relacionados con factores de personalidad, aunque pueden incluirse otros aspectos, como los planes de futuro personales o profesionales.

Los factores psicológicos juegan un papel muy importante en la motivación del trabajador. Resulta crítico mantener a los trabajadores motivados, puesto que se obtienen mejores resultados en su rendimiento profesional y evolución personal.

Los intereses personales también juegan un papel en este campo. Generalmente, cuanto mayor es el interés de un trabajador en aspectos de una tarea, mayor es su motivación para realizarla. Sin embargo, los intereses de cada trabajador pueden variar en gran medida con el transcurso del tiempo, dificultando su seguimiento y modelado. Estas variaciones se ven afectadas por una gran cantidad de aspectos: el ambiente de trabajo, situaciones concretas o vivencias personales.

Se decide no modelar estas características en el presente trabajo, ya que se considera que quedan fuera del alcance del mismo.

5.2 Análisis de tareas

El análisis de los trabajadores está orientado a establecer una relación del mismo con las tareas. Por lo que resultaría de escasa utilidad si no existe un estudio en profundidad de las tareas del proyecto.

El análisis de las tareas es fundamental para poder realizar una correcta asignación de los trabajadores. Los procedimientos orientados al análisis de tareas se suelen centrar en las actividades desarrolladas durante la realización del trabajo. Para un adecuado estudio se deben tener en cuenta los deberes, responsabilidades y funciones del trabajo.

Al igual que en el análisis del trabajador, existen diferentes sistemas que se pueden emplear. Uno de ellos es el análisis de tareas orientado cognitivamente (*COTA*), en él se observa a los trabajadores desarrollar las tareas para obtener información acerca de las habilidades y competencias requeridas en dicho puesto.

Este sistema resulta útil en las organizaciones con puestos estables, donde las tareas no varían a lo largo del tiempo. Estas se pueden analizar mientras son ejecutadas por un trabajador, para conocer las necesidades que implican. Este estudio se empleará posteriormente para seleccionar nuevos trabajadores para dicho puesto. Sin embargo, este sistema tiene carencias para la asignación de responsabilidades en proyectos. Aquí las tareas tienen una duración definida en el tiempo y, por lo tanto, no hay posibilidad de analizarlas antes del desarrollo del propio proyecto.

Otro sistema de análisis es el *FJA*, análisis funcional del trabajo. Se trata de una técnica que puntúa los diferentes elementos del trabajo con una escala numérica. La puntuación se

asigna en función de su complejidad en relación con los datos, las personas y las cosas. La ventaja de este método consiste en que no requiere que un trabajador esté realizando una tarea para poder analizarla.

Al analizar las tareas, y calificar las actividades involucradas, se puede lograr una mayor comprensión del trabajo. La información se puede organizar de forma similar al método KSAO empleado en el análisis de los trabajadores. Esto permite establecer una relación directa entre los atributos de los trabajadores y los requisitos de las tareas.

Las tareas en un proyecto cuentan con una serie adicional de características, debido al carácter temporal y lineal del proyecto: duración definida y relaciones de dependencia. Estas peculiaridades no afectan significativamente a la asignación de un trabajador, pero sí generan diferencias importantes que conciernen a la asignación de responsabilidades: la criticidad de las tareas.

5.2.1 Tareas críticas

Las tareas críticas son aquellas que pertenecen a la ruta crítica del proyecto. Esta se puede calcular mediante un análisis *CPM* (*Critical Path Method*). La ruta crítica es aquella que tiene la mayor longitud temporal en el proyecto y, por lo tanto, determina la duración total del mismo. Esto implica que cualquier retraso en una tarea de la ruta crítica afectará a la fecha de finalización del proyecto de igual modo.

Resulta conveniente asignar más o mejores recursos a las tareas de la ruta crítica, con el objetivo de evitar retrasos. Pero tampoco se deben ignorar del resto de tareas, especialmente si tienen una holgura temporal reducida, es decir, si su duración se aproxima a la duración de la ruta crítica. Si se emplean demasiados recursos en la ruta crítica, el resto de tareas puede sufrir retrasos debido a la falta de recursos. Esto puede alterar en la práctica la ruta crítica del proyecto debido a una mala gestión.

5.2.2 Conocimiento

Para realizar cualquier tarea correctamente es necesario disponer de unos conocimientos mínimos acerca de cómo llevar a cabo las actividades implicadas en la misma. En un puesto de trabajo estable esto no suele suponer un grave problema, ya que se puede dar una formación previa. Sin embargo, en un proyecto es necesario que los trabajadores ya dispongan de esa formación. De lo contrario se retrasaría todo el calendario del proyecto mientras se imparten los cursos de formación

Estos conocimientos requeridos se pueden definir de forma similar a aquellos de los trabajadores: engloban toda la información necesaria para poder realizar la tarea con éxito. Es posible definir de forma previa los conocimientos necesarios para llevar a cabo una tarea. Aun así, resulta recomendable que el responsable de determinar los conocimientos necesarios para cada tarea tenga experiencia previa en tareas similares. De esta forma, se puede evitar que los conocimientos exigidos sean excesivos o insuficientes; lo que generaría una asignación inadecuada.

Pese a que para cada conocimiento pueden considerarse diferentes niveles, vamos a considerar cada nivel como un conocimiento independiente. Esto facilitará la posterior automatización de la asignación de tareas al tratar de forma similar los conocimientos en las tareas y en los trabajadores.

5.2.3 Destrezas

Las destrezas son el conjunto de todas aquellas habilidades que es necesario dominar para poder desarrollar con éxito las actividades de una tarea determinada. En las tareas se pueden tratar las destrezas de forma similar al conocimiento: son imprescindibles para realizar correctamente el trabajo.

Al igual que los conocimientos, las destrezas se pueden definir de forma previa al progreso de la actividad. Aunque en este caso resulta aún más importante conocer el desarrollo de las actividades. Esto evitará que se incluyan destrezas innecesarias, o que se omitan algunas indispensables, en los requisitos de la tarea.

5.2.4 Competencias

El nivel competencial recomendable para desarrollar adecuadamente una tarea es más complejo de determinar que los conocimientos o las destrezas. Esto se debe a que viene determinado por diversos factores, algunos externos a la propia tarea. Por ejemplo, la dependencia de otras tareas, la falta de información, la modificación de objetivos o el ambiente de trabajo.

En cada tarea existe un determinado nivel de cada competencia que resultaría necesario tener para desarrollarla de forma óptima. Sin embargo, resulta inviable analizar todas las competencias para todas las tareas de un proyecto. Una solución consiste en destacar en cada proyecto las competencias que se consideren más importantes para su desarrollo, omitiendo el resto. Cada tarea del proyecto se evaluará de acuerdo a esas competencias.

Los métodos de evaluación de competencias en las tareas son diferentes a los usados para los trabajadores. En estos se emplean métodos de evaluación de competencias para asignar un valor numérico; sin embargo, para las tareas suelen utilizarse niveles competenciales. Estos niveles determinan en qué grado se espera que la competencia sea necesaria o recomendable. Se trata de un sistema más sencillo de cara a la evaluación. Aun así, es necesario haber establecido previamente los niveles competenciales y su definición. Este sistema tiene también la ventaja de que resultar fácil de transformar a una valoración numérica.

5.3 Métodos de asignación

Tras el estudio de los trabajadores y las tareas, es necesario analizar la relación existente entre ellos, y en qué modo afecta cada características al reparto de responsabilidades. Esto se empleará como guía para implementar el algoritmo de asignación automática.

Se ha decidido diseñar el algoritmo para que realice un proceso similar al de asignación manual. Se llevará a cabo un proceso de dos fases para determinar el trabajador idóneo.

En la primera fase se descartan los trabajadores que no son aptos para el trabajo. Para ello se comparan los conocimientos y destrezas de los trabajadores con aquellas requeridas por la tarea. Si el trabajador cuenta con los conocimientos y destrezas necesarias, se considera apto; de lo contrario se descarta. No se emplea ningún sistema de puntuación en este apartado.

De cara a la selección del trabajador para un puesto fijo, este primer paso no tendría por qué ser tan estricto, ya que se podrían cubrir las necesidades mediante un curso de forma-

ción. Sin embargo, en un proyecto es necesario que los trabajadores cuenten con los conocimientos y destrezas necesarios para llevar a cabo las actividades desde el principio.

Por otro lado, puede emplearse un método que tenga en cuenta los distintos grados de conocimiento o destreza. Permitiendo asignar también a trabajadores con niveles ligeramente inferiores o superiores. Sin embargo, para la demostración que se realizará más adelante, se considera que la aproximación de clasificar entre aptos y no aptos es aceptablemente correcta. Se considera que un método más complejo complicaría el algoritmo sin generar un beneficio proporcional, por lo que queda fuera del alcance del presente trabajo.

En la segunda fase de la asignación, tras analizar cuáles de los trabajadores son capaces de llevar a cabo la tarea de forma satisfactoria, es necesario investigar cuál de ellos la realizará de forma más eficiente. Esto se computa mediante una puntuación calculada de acuerdo a las competencias.

El análisis de competencias, tanto en los trabajadores como en las tareas, puede expresarse como una escala numérica. Esto facilita el procesamiento de la información y el uso posterior de algoritmos. La dificultad consiste en decidir el método idóneo para calcular la puntuación de desempeño de acuerdo a las valoraciones competenciales. Existen multitud de métodos que se podrían emplear.

Un primer método consiste en usar la media ponderada de las competencias de los trabajadores, empleando los pesos de las competencias en las tareas. Esto provocaría que, cuando una competencia sea muy relevante para una tarea, se valore mucho su nivel en los trabajadores de cara a la asignación.

Una variación de este primer método consiste en añadir penalizaciones a la puntuación si el nivel competencial de un trabajador se encuentra por debajo del nivel requerido por la tarea. Esto promovería que los trabajadores asignados siempre alcancen, o se acerquen considerablemente a los niveles mínimos deseables.

También se pueden establecer relaciones entre las competencias. Estas relaciones provocarían que la puntuación individual de una competencia pudiera verse ligeramente alterada por las otras competencias del trabajador. Este método podría servir para independizar las competencias evaluadas en las tareas y en los trabajadores; estableciendo relaciones entre competencias diferentes.

Entre las soluciones más avanzadas se podrían emplear métodos con inteligencia artificial para calcular las puntuaciones. En este apartado se pueden plantear dos enfoques: uno empleando lógica borrosa y otro empleando redes neuronales.

La lógica borrosa se adapta bien al mundo real en el que vivimos, e incluso puede comprender y funcionar con nuestras expresiones. Su principal ventaja es su capacidad para generar expresiones matemáticas y datos numéricos a partir de expresiones lingüísticas. Este sistema permitiría establecer relaciones conocidas entre las diferentes competencias, sin necesidad de introducir una valoración numérica manualmente.

En caso de emplear redes neuronales se podría emplear una capa de entrada con una neurona por cada competencia del trabajador y otra neurona por cada competencia de la tarea. Como salida tendría una única neurona con el dato numérico de la puntuación. Este método podría generar la salida de acuerdo a una función compleja, y no lineal, de las entradas, sin embargo, tiene varios inconvenientes.

En primer lugar, sería necesario entrenar la red neuronal con una gran cantidad de información difícil de obtener; no resultando útil hasta el momento en el que su entrenamiento se considere finalizado. Y, en segundo lugar, la red neuronal solo sería válida para una combinación fija de competencias. Para añadir o eliminar una competencia resultaría necesario modificar y volver a entrenar toda la red neuronal.

Todos los métodos pueden resultar válidos dependiendo de la situación y los objetivos de cada escenario. Por lo tanto, la decisión de implementar estos sistemas de puntuación, o cualquier otro, recaería sobre la organización. En el presente trabajo se va a calcular la puntuación como la media ponderada de los productos. Se considera que el resto de métodos añaden una complejidad que no resulta necesaria para alcanzar los objetivos del trabajo y que, por lo tanto, quedan fuera del alcance del mismo.

6. Demostración software

En este apartado se va a estudiar la viabilidad real de los modelos y del método de asignación planteado en el capítulo 5. Para ello se decide desarrollar una pequeña aplicación software. Esta aplicación empleará los modelos de los trabajadores y las tareas previamente mencionados e implementará los algoritmos para realizar las asignaciones de forma automática.

Al desarrollar esta aplicación y realizar pruebas con proyectos simulados se obtendrá una mayor información acerca de las limitaciones y funcionalidades de los modelos y métodos. Estos se emplearán de cara a una implementación real en una gran organización, que se estudiará en el capítulo 7.

6.1 Planteamiento de la aplicación

El primer paso para desarrollar una aplicación consiste en definir su estructura software: lenguaje, paradigma y herramientas. En este trabajo se considera que la mejor aproximación sería mediante la programación orientada a objetos, *OOP* (*Object-oriented programming*). Este paradigma de programación permitirá generar y administrar fácilmente una gran cantidad de elementos encapsulados en objetos junto con sus características individuales.

En la programación orientada a objetos la información acerca de un elemento se encapsula y abstrae para tratarla como un único bloque complejo, con atributos y métodos. Se trata de un paradigma que puede simplificar códigos más complicados, aportando características como el polimorfismo y la herencia. Sin embargo, no es recomendable en todas las ocasiones, pues también tiene sus inconvenientes. Entre ellos destacan la imposibilidad de programar empleando exclusivamente objetos y la necesidad de una amplia documentación de los objetos, debido a su abstracción frente al mundo real.

En el presente trabajo se considera que la programación orientada a objetos puede simplificar el desarrollo de la aplicación. Cada modelo se encapsulará en una clase, facilitando la gestión de una gran cantidad de elementos, cada uno de ellos representado en una instancia de la clase.

También se decide a desarrollar una interfaz gráfica de usuario (GUI) para el control de la aplicación, considerándose como un requisito de diseño. Esto se debe a que, en caso de una hipotética implantación en una organización, la aplicación debe disponer de un entorno gráfico adecuado para los todos usuarios. Además simplificará las pruebas realizadas, al emplear un entorno de trabajo más cómodo.

Finalmente, surge la necesidad de implementar una base de datos. En el caso de esta implantación demostrativa, su función será únicamente servir como herramienta para la carga y el guardado de datos. Sin embargo, en el caso de una implantación a gran escala, se trataría de una parte fundamental de la aplicación debido al mayor volumen de información.

6.2 Recursos empleados

Una de las decisiones más importantes antes de comenzar la programación de una aplicación consiste en decidir en qué lenguaje se va a desarrollar la misma. Cada uno tiene sus ventajas y sus inconvenientes, por lo que resulta muy recomendable elegir el lenguaje en función de la aplicación y del planteamiento inicial.

En este caso conviene escoger un lenguaje que facilite la programación orientada a objetos. Existen una amplia variedad de ellos, destacando: *C#*, *C++*, *Java*, *JavaScript*, *Python* y *Ruby*. Sin embargo, este no es el único requisito que debemos tener en cuenta en la elección del lenguaje.

Para implementar la interfaz de usuario es necesario emplear una librería gráfica. Por lo tanto, se deben tener en cuenta las diferentes opciones que existen para cada lenguaje de programación, y cuál se adapta mejor a las necesidades del proyecto.

La base de datos no forma parte de la propia aplicación, se trata de un elemento externo con su propio sistema de gestión. Sin embargo, la aplicación y el sistema de gestión de la base de datos tienen que interactuar para permitir el acceso a la información almacenada. Esto se suele realizar mediante librerías de los lenguajes de programación. La complejidad de implementación de estas interacciones y la funcionalidad de la base de datos puede variar entre librerías. Por lo tanto, a la hora de escoger el lenguaje de programación es necesario tener en cuenta el sistema de gestión de base de datos y la librería que se emplearán.

Valorando estos tres aspectos fundamentales de la aplicación (lenguaje, librería gráfica y base de datos) se toma una decisión. Se va a emplear el lenguaje de programación *Python*, junto a la librería gráfica *tkinter*. Para la implementación de la base de datos se usará el sistema de gestión de bases de datos *SQLite*, con la librería *sqlite3*.

Python es un lenguaje de programación interpretado multiparadigma con un buen soporte para la programación orientada a objetos. Se ha optado por este lenguaje frente a otros, como *C++* o *Java*, ya que permite un desarrollo más rápido, al tratarse de un lenguaje interpretado. Además, gracias a su reciente crecimiento, cuenta con una gran cantidad de documentación y librerías online. El planteamiento multiparadigma de *Python* otorga una mayor flexibilidad a la hora de plantear soluciones para cada problema.

Tkinter es una adaptación de la librería gráfica *Tcl/Tk*. Se considera la interfaz gráfica estándar para *Python*, instalándose por defecto junto al propio lenguaje. *Tk* es una aplicación y conjunto de controles destinados a desarrollar interfaces gráficas sencillas. Fue desarrollado para el lenguaje *Tcl*, este además se emplea como puente para utilizar la librería en *Python*. Se ha considerado como la mejor opción para desarrollar la interfaz gráfica, al contar con una buena integración con el lenguaje y amplia documentación.

SQL (*Structured Query Languaje*) es un lenguaje utilizado para programación, diseño y administración de bases de datos. *SQLite* es una librería multiplataforma que implementa un motor *SQL* para gestionar bases de datos. Su principal ventaja es que, al suprimir algunas funciones avanzadas, se consigue una gestión de la base de datos rápida y ligera. Se ha optado por emplear la implementación *sqlite3*, ya que se trata de un módulo estándar de *Python*, no siendo necesaria ninguna instalación adicional. Además, de forma similar a *tkinter*, al tratarse del módulo instalado por defecto, existe una amplia documentación online.

6.3 Modelado

Se decide diseñar la aplicación siguiendo el patrón modelo-vista-controlador (MVC). Este patrón separa en módulos independientes los datos, la gestión de los eventos y la representación visual. Para ello se construyen tres componentes distintos. En este caso el modelo, la información de los elementos implicados, se construye en el archivo *Library.py*; la vista se implementa en *Graphics.py*, y la gestión de ambos se realiza desde el controlador *Application.py*.

El modelo es el archivo donde se encuentran las definiciones de las clases de trabajador (*Worker*) y tarea (*Duty*). Estas dos clases contienen todos los atributos y métodos necesarios para la gestión de los trabajadores y las tareas. Si se requiriesen de funciones adicionales para gestionar los modelos, se implementarán en este archivo. Aunque, siempre que sea posible, los métodos se encapsularán dentro de una de las dos clases.

La vista es el archivo que contiene todas las funciones y clases necesarias para generar y controlar la interfaz gráfica. Por lo general, se emplea una función para controlar las ventanas cuyo único objetivo es mostrar información. En el caso de las ventanas interactivas, ya sea aquellas que retornan información, o las que requieren varias funciones para su control, se diseña una clase para encapsular todas sus funciones. Esto permite disponer de parámetros accesibles por todos los métodos de la ventana, pero no por el resto de la aplicación.

El controlador es el archivo que contiene todas las funciones necesarias para el correcto funcionamiento de la aplicación. Desde él se invocan las funciones de la librería gráfica y del modelo. Esto permite que las funciones principales de la aplicación se mantengan compactas y se encuentren desvinculadas de la implementación del modelo y de la interfaz.

La estructura de funcionamiento de este patrón se muestra en la figura 4.

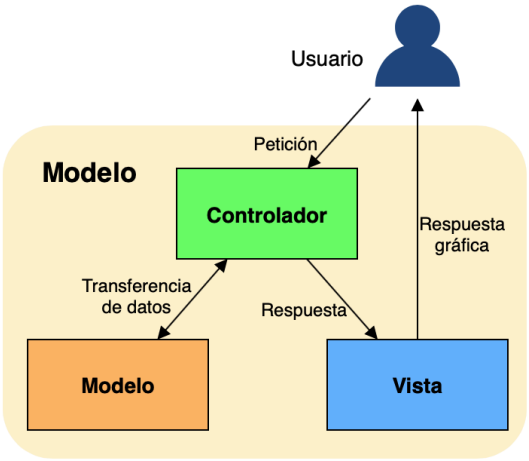

Fig. 4: Funcionamiento del patrón Modelo-Vista-Controlador

6.3.1 Trabajador

El modelo del trabajador se representa en la clase *Worker*. Esto permite encapsular todos los atributos y métodos del trabajador, a la vez que posibilita generar una gran cantidad de instancias, cada una con valores distintos, representando cada una a un trabajador real.

Los atributos de esta clase se pueden agrupar en tres categorías: datos identificativos, información de desempeño y responsabilidades en el proyecto.

Los datos identificativos son el nombre del trabajador (implementado como una cadena de caracteres), su apellido (también un *string*) y un número identificativo. La aplicación asigna este número de forma automática y se emplea para distinguir a dos trabajadores aunque sus nombres y apellidos sean idénticos.

Los parámetros relacionados con el desempeño se implementan siguiendo el método *KSAO* y el modelado de trabajadores. Los conocimientos y las destrezas se tratan de forma similar, actuando como un factor restrictivo a la hora de analizar si un trabajador es capaz

realizar una determinada actividad de forma efectiva. Cada conocimiento y destreza se implementa como una variable de tipo *string*. El conjunto de todos ellos se almacena dentro de una variable compuesta de tipo *set*. En *Python* los *sets* son conjuntos no ordenados de elementos inmutables no repetidos, por lo que resulta el tipo ideal para almacenar el listado de conocimientos y destrezas de los trabajadores.

Las competencias se almacenan en una variable de tipo diccionario. Los diccionarios de *Python* son colecciones no ordenadas de datos mutables accesibles mediante una clave inmutable. El nombre de cada competencia actúa como clave y su valoración es el dato almacenado. Esto permite acceder fácil y rápidamente a la valoración de cada competencia. Además, las claves son inmutables y no permiten repetición, por lo que cada competencia solo puede tener una valoración. En caso de tratar de guardar dos veces la misma competencia, tan solo se almacena el valor más reciente.

Finalmente se encuentran los atributos relacionados con el cometido del trabajador dentro del proyecto. En primer lugar se usa una variable de tipo lista (conjunto ordenado de variables mutables) para almacenar los punteros a las tareas asignadas al trabajador. Y, en segundo lugar, se emplea otra lista para controlar los días en que el trabajador se encuentra ocupado. Esta lista funciona almacenando de forma indistinta los días de inicio y fin de cada actividad. Puesto que cada trabajador no puede realizar dos actividades a la vez, las posiciones pares de la lista representan los inicios de las actividades y las posiciones impares representan los finales.

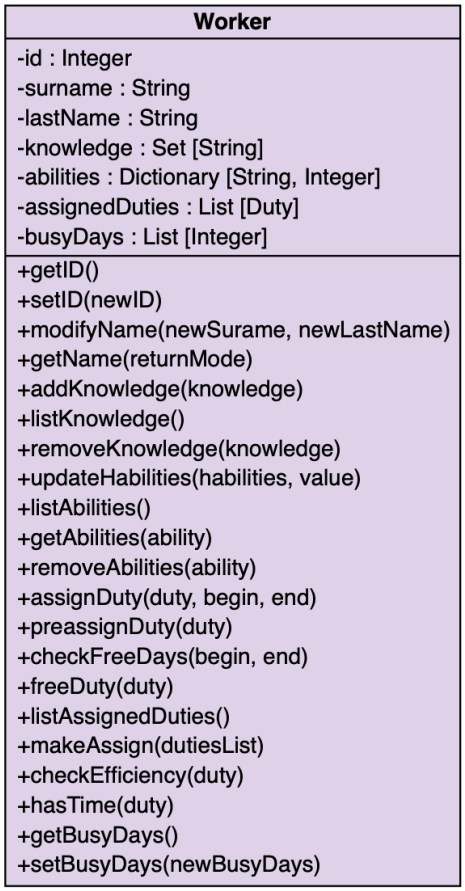

Fig. 5: Clase del trabajador

Para obtener o modificar la información guardada se emplean siempre métodos específicos de la clase. Python no permite la privacidad de los métodos o los atributos, aun así, no

resulta conveniente acceder directamente a los atributos. Esto aumenta la cantidad de métodos de la clase, pero permite implementar seguridad, evitando guardar datos incorrectos en cualquiera de los campos. Esto último puede ser un problema especialmente peligroso en *Python*, puesto que no se pueden especificar manualmente los tipos de las variables empleadas.

Además se implementan métodos más específicos que permiten, por ejemplo, conocer la valoración del trabajador para una determinada tarea, o comprobar si entre dos fechas se realiza alguna tarea o el trabajador está libre en dicho intervalo. La clase del trabajador, junto con todos sus atributos y métodos, se puede observar en la figura 5.

6.3.2 Tarea

El modelo de la tarea se implementa de forma similar al del trabajador. Se encapsula dentro de una clase, denominada *Duty.* Esta clase cuenta con una mayor cantidad de atributos y métodos para poder almacenar y gestionar la información relevante de cada tarea dentro del proyecto.

De forma similar al trabajador se pueden clasificar los atributos en tres categorías: datos identificativos, datos de desempeño y datos relacionados con el proyecto.

En los atributos identificativos se implementa una variable numérica de tipo entero que sirva como número identificativo de la tarea, de forma similar a la clase *Worker*. Para permitir una rápida identificación de cada tarea por parte del usuario, se les asigna un atributo de tipo string que contenga el objetivo de la tarea. Este actúa de modo similar al nombre de los trabajadores.

Para valorar el desempeño de un trabajador en la tarea se implementan dos parámetros similares a los que se implementaron en la clase *Worker*. En primer lugar se almacenan los requisitos de la tarea en un *set*. Este atributo contendrá requisitos de conocimientos y destrezas necesarias para llevar a cabo la tarea, cada uno de ellos representado como una variable de tipo *string*. Esto permite relacionarlo y compararlo fácilmente con el *set* de conocimientos y destrezas de los trabajadores. Las competencias también se implementan de forma similar a la clase *Worker*, mediante un diccionario donde el nombre es la clave y el dato es el nivel competencial asignado.

Los atributos utilizados para relacionar las tareas con el proyecto difieren respecto a las empleadas en el caso del trabajador, siendo más numerosas en este caso. En primer lugar, a cada tarea se le asigna una duración estimada. Este parámetro afectará a la distribución temporal de las tareas dentro del proyecto

También se implementan dos atributos de tipo *set* para gestionar las dependencias entre tareas. Dentro de estos dos conjuntos se almacenarán los punteros a las tareas previas y posteriores respectivamente. La información de las dependencias se podría almacenar empleando únicamente uno de los dos conjuntos; sin embargo, este método facilita el control sobre las relaciones de tareas, pudiendo recorrer el proyecto en ambos sentidos.

Con las dependencias entre tareas y la duración de cada tarea se pueden calcular las fechas de inicio mínimo y máximo de cada tarea. Ambas, junto con la fecha del inicio previsto de la tarea, se almacenan en un atributo de tipo diccionario. Empleando estas fechas se puede calcular si la tarea tiene holgura en sus fechas de inicio o si, por el contrario, se trata de una tarea perteneciente a la ruta crítica del proyecto.

Con el fin de facilitar la gestión posterior de las tareas, se implementa una variable de tipo booleano que sirve para conocer si la tarea forma parte de la ruta crítica. Esto evita tener que realizar el cálculo cada que vez que se quiera conocer esta información. Finalmente se implementa un último atributo que sirve para identificar al trabajador que se encuentra asignado a esta tarea, por lo tanto se trata de una variable de tipo puntero a clase *Worker*.

Al igual que en la implantación de la clase del trabajador, se accede a todos los atributos mediante métodos. Esto evita que puedan introducirse datos incorrectos en cualquiera de los campos de la clase.

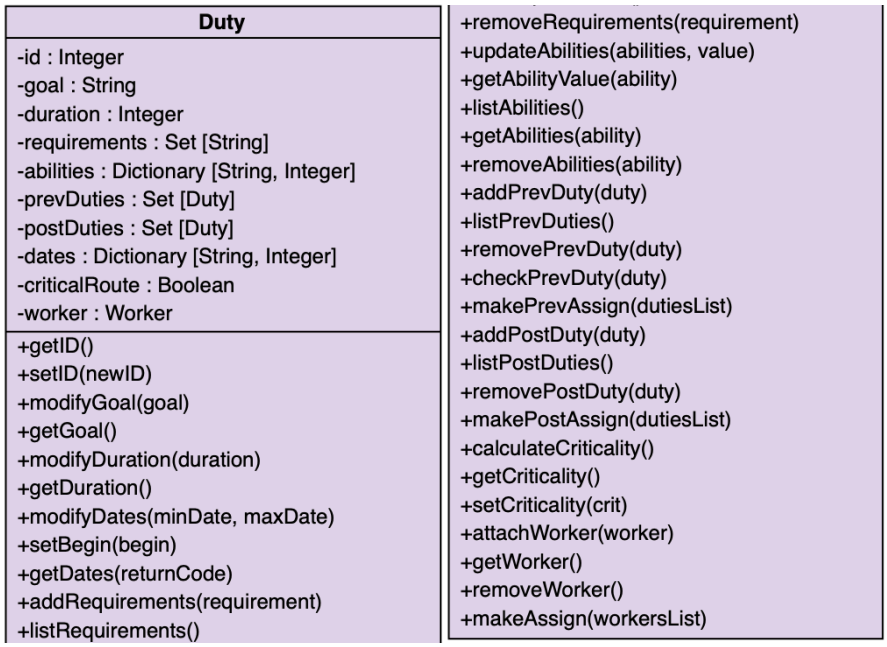

Fig. 6: Clase de la tarea

6.3.3 Aplicación

El usuario debe poder acceder a todas las funcionalidades de la aplicación mediante la interfaz gráfica. Por lo tanto, esta debe diseñarse de acuerdo a las acciones que deseamos que pueda llevar a cabo el usuario. Para ello se realiza una diagrama simplificado de casos de uso, mostrado en la figura 7.

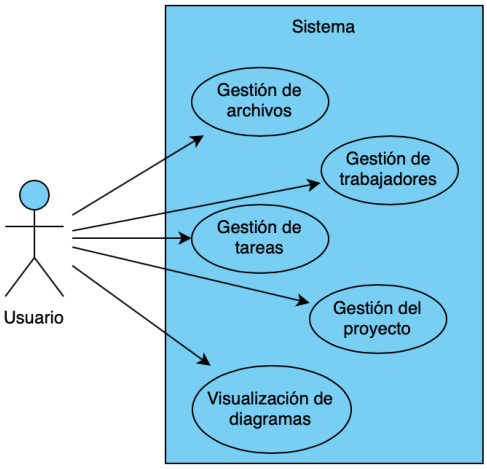

Fig. 7: Diagrama simplificado de casos de uso

La interfaz gráfica comienza con una ventana principal desde la que se puede acceder a las distintas funcionalidades mediante el uso de botones. Estas se pueden clasificar en cuatro categorías: gestión de archivos, gestión de recursos (trabajadores y tareas), gestión del proyecto y visualización de diagramas.

En la primera categoría se encuentran los botones encargados de gestionar los archivos de los proyectos. Se emplea un botón para guardar los proyectos, que no generará ninguna ventana adicional, y otro para cargar los proyectos, que abre otra ventana para seleccionar el nombre del nuevo proyecto.

Tras estos botones, en un segundo grupo se disponen aquellos relacionados con la gestión de los recursos del proyecto activo: los trabajadores y las tareas. Para cada uno de ellos existen tres interacciones.

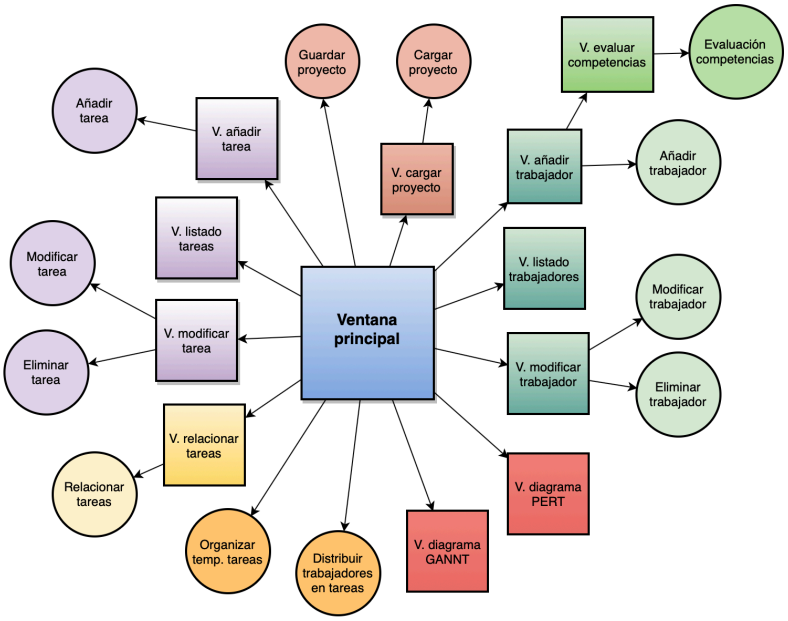

Fig. 8: Jerarquía de ventanas y acciones

Es necesario poder añadir nuevos recursos al proyecto; para ello se implementa un botón que abre una nueva ventana. Esta contiene un formulario donde rellenar los datos identificativos y los relacionados con el desempeño del recurso en cuestión. Para que la valoración de las competencias en los trabajadores sea adecuada, se puede acceder a una ventana auxiliar de evaluación de competencias. Esta sirve para obtener una valoración numérica a partir de respuestas a un formulario predefinido.

También se debe permitir la visualización de la información de los recursos de forma rápida. Para ello se emplea una ventana donde se muestran todos los trabajadores, o tareas , junto con sus atributos identificativos, de desempeño y del proyecto.

Se deben poder realizar modificaciones sobre los recursos. La ventana asignada a tal fin permite seleccionar un recurso y escoger entre editarlo, generando un formulario similar al de la creación, o eliminarlo completamente del proyecto.

Un tercer grupo de botones proporciona acceso a las funciones de gestión del proyecto. Estas permiten definir las relaciones de dependencia entre tareas, determinar la previsión temporal del proyecto y generar la asignación de los trabajadores a las tareas. La primera de

estas funciones, la generación de dependencias, requiere de una interacción del usuario, por lo que es la única que abrirá una nueva ventana.

El cuarto, y último, grupo de botones está destinado a la representación gráfica del proyecto. Se ha decidido emplear los diagramas GANNT y PERT; el diagrama GANNT se emplea para conocer la distribución temporal de las actividades, mientras que el diagram PERT se utiliza para estudiar la ruta crítica del proyecto y las dependencias entre tareas. Se considera que se trata de dos diagramas complementarios, y de gran utilidad para cualquier proyecto. En ninguno de ellos resulta necesaria la interacción del usuario, aunque sí es indispensable generar una nueva ventana donde se representará el diagrama.

En la figura 8 se muestran todos los elementos de la aplicación. Los cuadrados representan las ventanas, mientras que los círculos representan las acciones que se pueden llevar a cabo desde cada tarea.

6.4 Algoritmos

Una vez se han modelado los elementos fundamentales de la aplicación y se ha definido la jerarquía de ventanas y funcionalidades, es preciso diseñar los algoritmos que se encargan de las tareas computacionalmente más complejas. No se van a exponer todos los algoritmos implementados en la aplicación, pero sí se juzga adecuado explicar aquellos que se consideran más importantes.

Se ha estimado que los algoritmos más relevantes en el presente trabajo son los encargados la asignar las fechas a las tareas, el reparto las tareas entre los trabajadores y la representación el gráfico PERT.

6.4.1 Asignación Fechas

La asignación de fechas solo se puede realizar tras definir las relaciones de dependencias entre tareas. Para facilitar algunos cálculos se generan dos tareas auxiliares de duración instantánea: tarea de inicio y tarea de final. Estas actúan como punto de partida para los algoritmos.

Una tarea solo puede comenzar cuando todas las tareas previas han finalizado, y debe terminar antes de que suponga un retraso para la fecha de finalización del proyecto. Estas son las dos fechas clave para el cálculo temporal del proyecto: la fecha de inicio mínimo y la fecha de inicio máximo. La fecha de comienzo real depende de la ocupación del trabajador asignado, pero debe situarse siempre entre el inicio mínimo y el máximo.

Este algoritmo se puede dividir en cuatro fases claramente diferenciadas: preparación, cálculo del inicio mínimo, cálculo del inicio máximo y cálculo de la ruta crítica.

Preparación

Para asignar las fechas se comienza realizando un barrido por todas las tareas para asegurar que las relaciones de dependencia están bien establecidas. Todas las tareas, excepto el inicio y el final, deben contar con tareas previas y posteriores. Si una tarea no tiene tareas previas, se le asigna como tarea previa la tarea auxiliar de inicio; y, si no tiene tareas posteriores, se le asigna como tarea posterior la tarea auxiliar de final. Con esto se logra tener un inicio y un final únicos en el proyecto.

A continuación se fijan las fechas de inicio mínimo y máximo de la tarea de inicio el día 1 del proyecto. El resto de tareas se fijan con fecha de inicio mínimo el día 0.

En la figura 9 se muestra un ejemplo de como se realiza este proceso para una de las tareas. Este proceso se realiza mediante un bucle con todas las tareas.

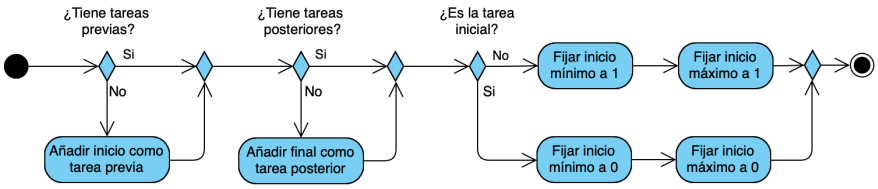

Fig. 9: Preparación para el cálculo de fechas de una tarea

Inicio mínimo

La fecha de inicio mínimo se define como el máximo de los finales mínimos de las tareas previas. Por lo tanto, el final mínimo es el inicio mínimo más la duración. Para cada tarea se realiza un barrido por sus tareas previas, calculando sus finales mínimos. Si el final mínimo calculado es mayor al inicio mínimo de la tarea estudiada, se actualiza el final mínimo por el nuevo valor calculado.

Para asegurar que la asignación está completa, y que ningún valor ha quedado mal calculado o corrupto por algún motivo, se repite el cálculo de los inicios mínimos de todas las tareas hasta que en una iteración completa no se realiza ninguna modificación de fechas. Esto implica que dos iteraciones seguidas han obtenido el mismo resultado, dándose por válida la asignación. El proceso seguido para cada una de las tarea del proyecto se muestra en la figura 10.

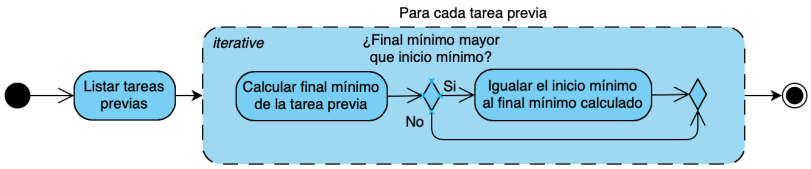

Fig. 10: Cálculo de los inicios mínimos de una tarea

Inicio máximo

La fecha de inicio máximo se define como el mínimo de los inicio máximos de las tareas posteriores menos la duración de la tarea actual. Para poder calcular de forma adecuada el inicio máximo, se debe comenzar por fijar todos los inicios máximos de las tareas del proyecto a la fecha de finalización del proyecto. De esta forma se pueden realizar correctamente las comparaciones entre fechas del algoritmo. Esta fecha final del proyecto se puede obtener de la tarea auxiliar de final puesto que no tiene duración ni holgura, por lo que su inicio mínimo, inicio máximo y final mínimo y final máximo coinciden entre sí y con la finalización global del proyecto.

El cálculo sigue una estructura similar a la de los inicios mínimos. Para cada tarea se realiza un barrido de sus tareas posteriores. Si el inicio máximo de la tarea posterior menos la duración de la tarea actual es menor al inicio máximo de la tarea actual, se actualiza su valor por el calculado. Este proceso también se repite hasta que una iteración completa no realice ninguna modificación. El proceso que sigue cada tarea se muestra en la figura 11.

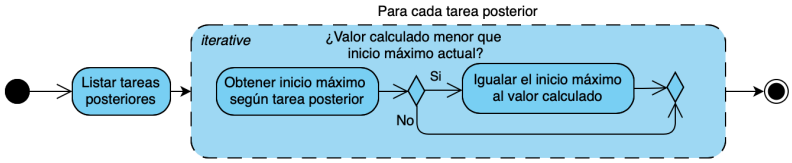

Fig. 11: Cálculo de los inicios máximos de una tarea

Ruta crítica

El último paso que realiza este algoritmo consiste en calcular las tareas que forman parte de la ruta crítica. Para ello se efectúa un nuevo barrido sobre todas las tareas invocando un método de la clase *Duty* que calcula su criticidad. Una tarea es crítica si no tiene holgura entre su inicio mínimo y su inicio máximo. Por lo tanto se realiza una comparación entre ambos valores y, según el resultado, se asigna el valor adecuado al atributo booleano que representa la criticidad.

Con esto queda finalizado el reparto temporal de las tareas del proyecto. El flujo de trabajo del algoritmo se muestra en la figura12.

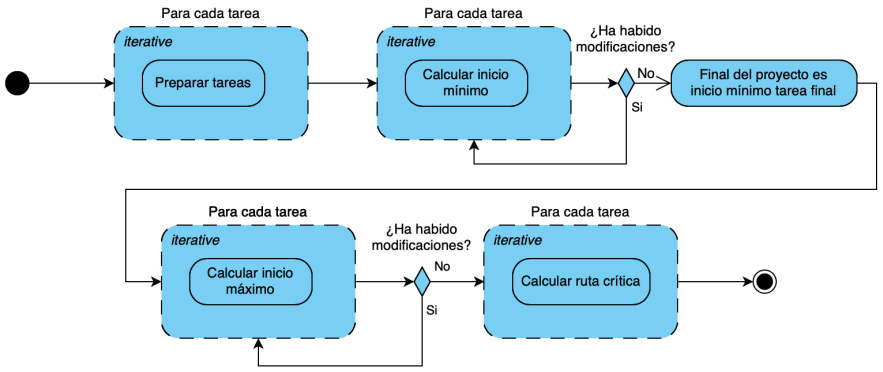

Fig. 12: Algoritmo de asignación de tiempos

6.4.2 Reparto de trabajadores

La asignación de los trabajadores se debe realizar tras el cálculo de fechas del apartado anterior. Esta asignación cuenta con dos fases al discriminar la prioridad de asignación de las tareas en función de su pertenencia a la ruta crítica. En una primera etapa se asignan los trabajadores óptimos a las tareas de la ruta crítica. Al priorizar estas, se pretende evitar los retrasos por falta de recursos. Esto es de gran importancia ya que cualquier retraso en una de estas tareas afectaría a la fecha de conclusión del proyecto.

Una vez las tareas de la ruta crítica se encuentren asignadas, se procede a asignar el resto de tareas. Es probable que estas no las lleve a cabo el trabajador mejor cualificado para las mismas, puesto que estará realizando una tarea de la ruta crítica. Sin embargo, como en esta ocasión existe holgura en las fechas, se considera aceptable.

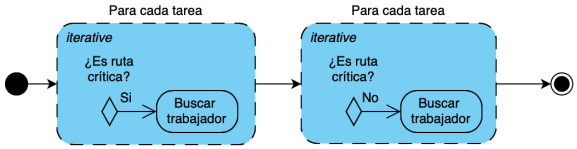

Fig. 13: Prioridad de las tareas de la ruta crítica

En ambas etapas la asignación se realiza de forma similar, por lo que se emplea una misma función. Esta recibe el puntero de la tarea que se desea asignar, y comienza comprobando si las tareas previas a la recibida tienen un trabajador asignado. En caso contrario, actúa como una función recursiva, llamándose a sí misma y realizando el paso por referencia de la tarea sin trabajador. Esto pretende evitar problemas de tiempos en la asignación de tareas, determinándolas siempre desde el inicio del proyecto hacia el final.

Cálculo de la puntuación de desempeño

Durante esta fase se procede a buscar un trabajador idóneo para la tarea, para ello se realiza un barrido por los trabajadores del proyecto. Para cada uno de ellos se deduce una puntuación de desempeño. Esta puntuación la calcula un método de la clase *Worker* al pasarle por referencia la tarea en cuestión.

En primer lugar este método comprueba que entre los conocimientos del trabajador se encuentren todos los requisitos de la tarea. En caso contrario devuelve un número negativo indicando que no puede hacerse cargo de la tarea.

Si el trabajador puede asumir la tarea, se calcula la puntuación de desempeño. Para ello realiza el producto entre la valoración de cada competencia en el trabajador y en la tarea, y suma todas las puntuaciones individuales de las competencias entre sí. El resultado devuelto es un valor numérico que representa su desempeño estimado en la tarea.

De vuelta en la función principal, la puntuación obtenida se compara con una puntuación guardada, que comienza siendo 0. Si la puntuación del nuevo trabajador supera la guardada, se considera que es mejor que la anterior asignación. En este punto es necesario conocer si el trabajador dispone de tiempo para realizar la tarea.

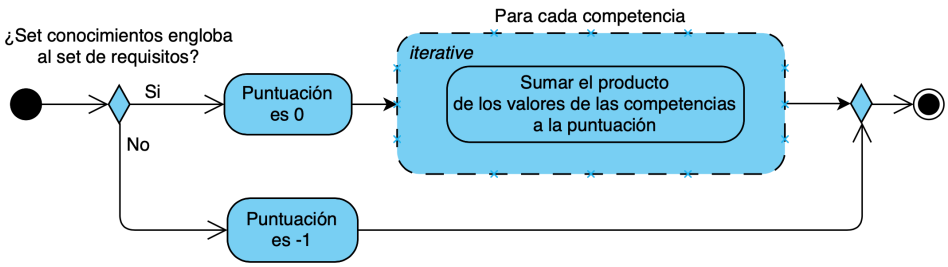

Fig. 14: Cálculo de puntuación de desempeño

Búsqueda de fecha

Para buscar una posible fecha de realización de la tarea se emplea un método de la clase *Worker.* Este comienza por determinar la fecha de inicio mínimo real de la tarea de acuerdo a sus tareas previas. Esta fecha es el máximo entre el inicio mínimo de la tarea y los finales reales de las tareas previas, calculados como su inicio real más su duración. En este punto resulta de gran importancia que todas las tareas previas cuenten con un trabajador y una fecha de inicio real asignados. De lo contrario podrían aparecer problemas o incongruencias en la asignación temporal de las tareas. Por ello es importante la primera fase, en la que el algoritmo comprueba todas las tareas previas.

Desde esta fecha de inicio mínimo real, y hasta la de inicio máximo, se itera en cada día. El método comprueba si el trabajador tiene libre el tiempo necesario para llevar a cabo la

tarea. Esta comprobación se realiza en otro método de la clase *Worker* que recibe como parámetros de entrada la fecha de inicio y la fecha de final deseadas.

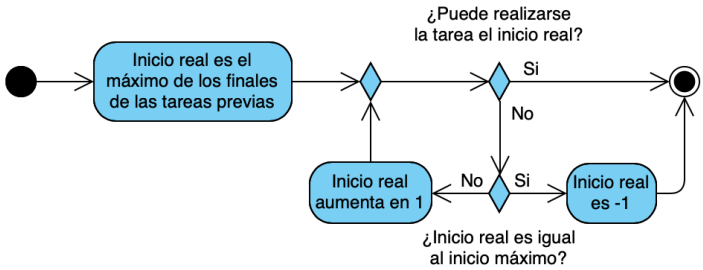

Fig. 15: Cálculo de la fecha de inicio real

El método de comprobación de fechas compara las entradas con la lista de días ocupados del trabajador, devolviendo un resultado positivo cuando se cumple una de las cuatro condiciones necesarias para que el trabajador pueda realizar la tarea:

- La lista está vacía (ningún día ocupado).
- El final es menor al primer valor de la lista (será la primera tarea en realizarse).
- El inicio es mayor al último valor de la lista (será la última tarea en realizarse).
- El inicio es mayor a cualquier valor par de la lista, a la vez que el final es menor al siguiente valor (la tarea se realiza entre otras dos tareas).

En caso de no cumplirse ninguna de estas cuatro condiciones, se devuelve un resultado negativo.

Cuando una de las comprobaciones es positiva, se finalizan ambos métodos y se devuelve al algoritmo de asignación la fecha escogida. Esta será la más temprana en la que el trabajador puede realizar la tarea.

Asignación del trabajador

Si la nueva fecha es válida (en caso contrario será negativa) se guarda el trabajador , provisionalmente, como el nuevo trabajador óptimo. También se guardan su puntuación de desempeño, de cara a las futuras comprobaciones, y la fecha calculada, con vistas a una asignación definitiva.

Cuando el barrido de los trabajadores ha finalizado, se procede a asignar de forma definitiva el trabajador escogido a la tarea, empleando la fecha guardada como la fecha de inicio. El proceso completo se puede observar en la figura 16.

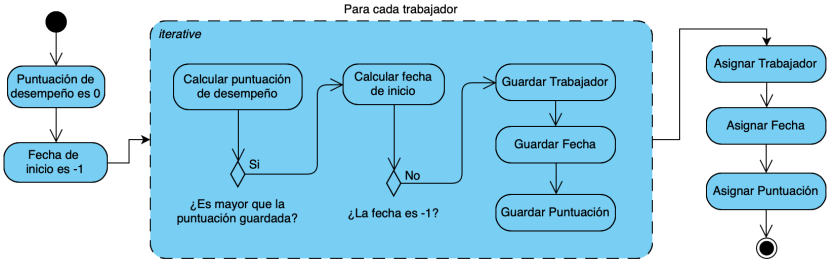

Fig. 16: Asignación de trabajador para cada tarea

Una vez se ha completado este proceso para todas las tareas, todas ellas deberían contar con un trabajador asignado. Si una tarea no dispone de un trabajador asignado quiere decir que el proyecto no puede completarse en el tiempo mínimo con los recursos actualmente a su disposición.

6.4.3 Diagrama PERT

Una malla PERT (*Program Evaluation and Review Techniques*) permite planificar y controlar el desarrollo de un proyecto. Para dibujar una malla PERT es necesario conocer la duración, el inicio mínimo y el inicio máximo de cada tarea. Estos datos ya están calculados, por lo que no supondrán ningún problema. Sin embargo, existen dos obstáculos para representar gráficamente la malla.

Se ha decidido realizar la malla PERT con actividad en los arcos, y esto trae el inconveniente de que se debe hacer una transformación de los datos para adaptarlos a las nuevas necesidades. En la transformación es necesario advertir que los nodos tienen que ser únicos y que cada tarea solo puede aparecer una vez en la malla. En caso de necesidad habrá que generar tareas virtuales para que la malla sea adecuada.

El segundo problema aparece con la representación gráfica, ya que se debe dibujar el gráfico con el inicio del proyecto a la izquierda de la pantalla, y el final a la derecha, evitando cruces o solapamientos de tareas. La solución empleada para resolver este problema consiste en construir una matriz con los nodos, que posteriormente se representará en la pantalla.

Transformación a nodos

El primer paso consiste en transformar la lista de tareas a una lista de nodos. Cada nodo es el final de unas tareas y el inicio de otras. Para modelar los nodos se decide emplear diccionarios de dos posiciones: una representando las tareas que comienzan y otra representando las tareas que finalizan; todos los nodos se almacenan en una lista. Podría resultar más conveniente almacenarlos en un *set*, ya que se eliminarían automáticamente los nodos duplicados, pero los diccionarios son tipos de datos mutables, por lo que no pueden almacenarse en el *set*.

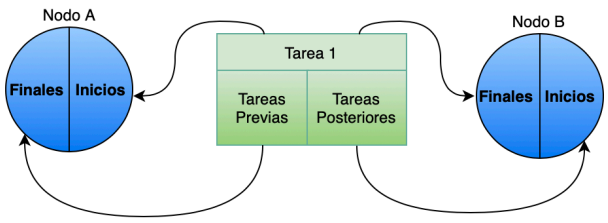

Fig. 17: Representación de la transformación de tareas a nodos

En primer lugar se emplea una función que genera nodos a partir de una lista de tareas. Cada tarea analizada se convierte en la tarea que parte del nodo. Mientras que las tareas previas a la misma se convierten en las tareas que finalizan. Para asegurar completamente que todas las tareas se transforman, este método se llama a sí mismo con cada una de las tareas previas o posteriores de la tarea analizada.

Sin embargo, cada tarea transformada ha sido previamente colocada en una lista que se comprueba al comienzo del método. Esto evita generar más de un nodo a partir una misma tarea, y también previene la aparición un bucle recursivo sin fin.

El siguiente paso consiste en asegurar que no existen nodos duplicados. Para ello se realiza un doble barrido por todos los nodos. En él se compara cada nodo con todos los demás. Si aparece una coincidencia, si hay dos nodos idénticos, uno de los dos nodos se elimina.

Tras eliminar los duplicados, sigue habiendo nodos con las mismas entradas (tareas que finalizan) y distintas salidas (tareas que comienzan). Se trata de tareas con las mismas dependencias previas, por lo que en la red se trata de un único nodo del que parten todas las tareas salidas. Por lo tanto, es necesario realizar nuevamente un doble barrido de todos los nodos buscando aquellos que comparten las entradas. En el caso de encontrar alguna coincidencia, se combinan las salidas y se elimina uno de los dos nodos. El resultado se muestra en la parte derecha de la figura 18.

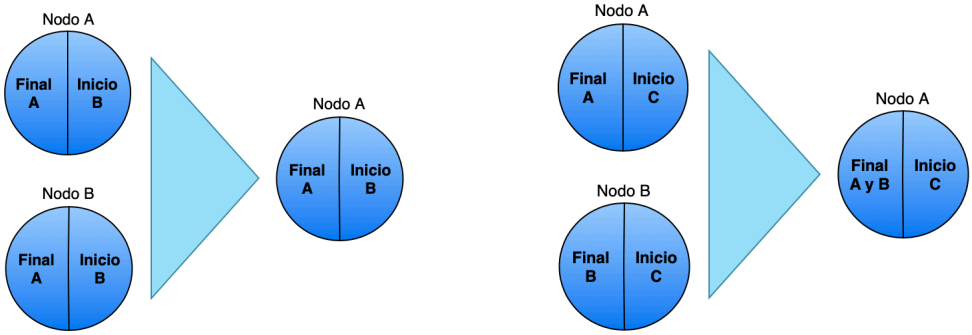

Fig. 18: Representación de la eliminación de duplicados (izq.) y combinación de nodos (der.)

En una malla PERT puede resultar necesario generar una tarea virtual para representar correctamente las relaciones entre tareas. Esto ocurre cuando dos nodos comparten parte de las tareas que finalizan, pero no todas. Para hacer una correcta representación es necesario buscar los nodos que requieren de tareas virtuales. Por lo tanto, se realiza nuevamente un doble barrido de los nodos, en esta ocasión buscando partes coincidentes entre las salidas de los estados. La parte coincidente en el nodo que tiene más salidas, es la que debe convertirse en una tarea virtual. Se debe sustituir esta parte por una tarea virtual, y añadirla además, al otro nodo como entrada. Este proceso se muestra gráficamente en la figura 19.

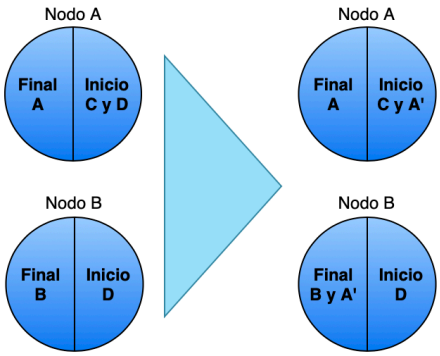

Fig. 19: Representación de la generación de una tarea virtual

Preparación de la matriz de nodos

Una vez generados los nodos y las tareas virtuales, la transformación de tareas a nodos se da por finalizada. El siguiente paso consiste en preparar los nodos para ser representados. Para ello se genera una matriz en la que se distribuyen los nodos. A fin de representar todos los nodos, evitando posibles duplicidades, se opta por generar una lista con todos

ellos. Según se introducen en la matriz, se eliminan de la lista; cuando la lista se encuentre vacía, la matriz estará finalizada.

Se comienza generando una matriz de una fila y una columna donde se dispone el nodo inicial, en el que finaliza la actividad de inicio. Tras esto se añade una nueva columna a la matriz y se genera una lista con las tareas que comienzan pero no acaban en la matriz. y se eliminan aquellas tareas que ya finalizan dentro de la matriz. A esta lista hay que eliminarle aquellas tareas que no tengan todas sus tareas previas en la matriz. Todo esto se consigue mediante varios barridos a la lista de tareas pendientes de poner en la matriz. El resultado es la lista de nodos que hay que añadir a la matriz en la nueva columna.

A continuación hay que determinar en qué fila se dibujará el nodo. Esta se calcula como la media de las filas de los nodos con tareas en común. En caso de estar esa fila ocupada, se añade una nueva fila a la matriz. Con esto se pretende minimizar los cruces entre tareas en el diagrama.

Este proceso se repite de forma continua, añadiendo en cada iteración una nueva columna con nodos. El bucle finaliza cuando la lista de nodos pendientes se encuentra vacía.

En la figura 20 se muestra gráficamente este método. En cada paso se añade una columna y las filas necesarias para alojar los nuevos nodos, aunque puede no ser necesaria ninguna fila. En esta representación se han marcado los cuadros vacíos de la matriz con aspas.

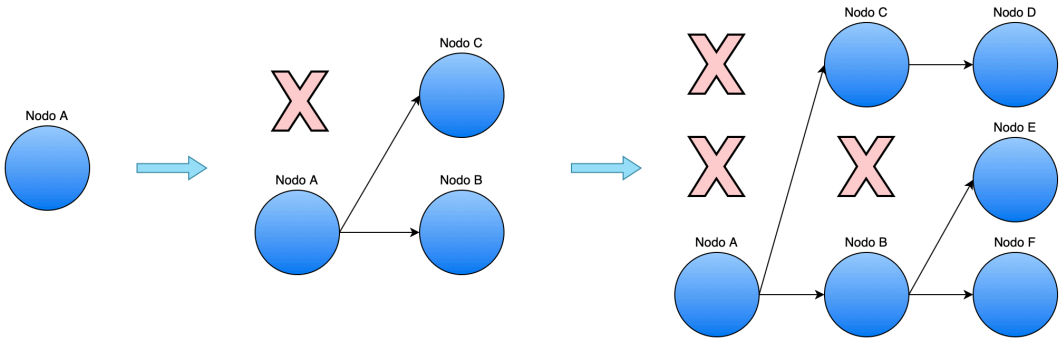

Fig. 20: Representación del desarrollo de la matriz

Dibujo de la matriz

El tamaño y la forma de los nodos esta predefinida; sin embargo, su el punto en el que se representará viene determinado por la posición que ocupa dentro la matriz. Por lo tanto, se recorre la matriz, relacionando directamente la columna en la matriz con la posición horizontal, y la fila con la posición vertical. Se realiza un barrido en las dos dimensiones de la matriz hasta dibujar todos los puntos.

Posteriormente se realiza otro barrido con el objetivo de dibujar las flechas de la malla PERT. Para ello se buscan nodos que compartan tareas, en un caso como entradas y en el otro como salidas. La posición de estos nodos en la matriz define los puntos de inicio y final de la flecha en el gráfico. Para mejorar la calidad y la información representada, se analiza si la tarea es virtual o crítica, modificando su representación gráfica en estos casos.

Finalmente se puede emplear la información acerca de los tiempo mínimos y máximos para añadir información al gráfico.

El flujo de trabajo para generar el diagrama PERT se muestra en la figura 21:

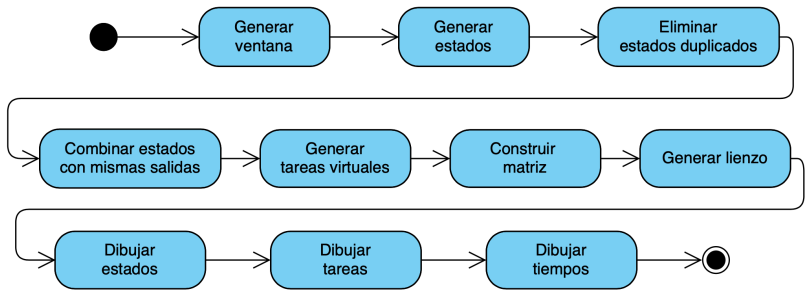

Fig. 21: Pasos necesarios para el diagrama PERT

6.5 Base de datos

La primera base de datos está orientada al guardado y cargado de proyectos. Esta va a ser su única función, puesto que la implementación en profundidad de una base de datos completa con el seguimiento del proyecto queda fuera del alcance de este trabajo.

También se va a emplear otra base de datos para el cargado de la información acerca de las competencias. Esto permite que la información de las competencias, así como sus métodos de evaluación, no se encuentren dentro del código del programa, sino en un archivo independiente.

Base de datos de competencias

La base de datos relativa a las competencias contiene una tabla donde se encuentran todas las competencias. Cada competencia emplea una tabla adicional, donde se almacena la información necesaria para llevar a cabo la evaluación competencial. En cada tabla de competencia se encuentran las preguntas de evaluación, la lista de respuestas a dicha pregunta y la valoración numérica para cada respuesta.

Al arrancar el programa se carga automáticamente la base de datos de competencias que se encuentra en la misma carpeta que la aplicación. Esto permite cambiar fácilmente entre distintas combinaciones de competencias, sin necesidad de modificar el código, tan solo es necesario introducir en la carpeta del proyecto la base de datos con las competencias deseadas.

En caso de no cargar ninguna base de datos de competencias, o no cargar la adecuada, las competencias de los trabajadores no se ven afectadas, como tampoco las asignaciones de tareas. Sin embargo, la aplicación no podrá llevar a cabo la evaluación ni la valoración de competencias.

Base de datos de proyectos

Se emplea una base de datos independiente para gestionar los proyectos. Cada proyecto se almacena una base de datos independiente. Desde el botón de carga de proyectos de la ventana principal de la aplicación se puede cargar una base de datos determinada. Si no se encuentra la base de datos especificada, se genera un nuevo proyecto, junto con la base de datos correspondiente, con el nombre indicado.

Esta base de datos contiene dos tablas: una de trabajadores y otra de tareas. En ellas se guarda toda la información contenida en los atributos de los objetos. Cada fila es un elemento (trabajador o tarea), y cada columna es un atributo del elemento. En el caso de los atributos complejos (sets, listas y diccionarios) se introducen en una misma celda separados por

comas a modo de cadena de caracteres. El método de extracción de información se encarga de reconvertir estos *strings* recuperados al formato necesario.

El principal problema al trabajar con estas tablas surge al recuperar la información que, en las clases, se almacena como un puntero a otro objeto. Esto se debe a que los punteros están asociados a las direcciones de memoria de los objetos, y estas direcciones cambian de una sesión a otra.

La solución implementada consiste en no guardar el propio puntero, ya que este no será el mismo una vez cargada la información, sino el identificador de la tarea o trabajador a la que apunta. Gracias al sistema de gestión de variables de *Python*, se pueden cargar todos los objetos en la memoria del programa sin importar el tipo de dato. Posteriormente, se realiza una barrido por las tareas y los trabajadores sustituyendo los identificadores obtenidos de la base de datos por los punteros correspondientes en la memoria de la aplicación.

6.6 Resultado de la aplicación

Seguidamente se va repasar el resultado final de la aplicación. Se va a exponer la interfaz, mostrando las diferentes ventanas, y las acciones que puede ejecutar el usuario.

6.6.1 Ventana Principal

Fig. 22: Ventana Principal de la aplicación

La ventana principal de la aplicación no muestra información acerca del proyecto, tan solo cuál es el proyecto activo. Como esta ventana sirve de punto de partida para acceder al resto de funcionalidades, se ha desarrollado siguiendo el modelo propuesto originalmente en el apartado *6.3.3*.

6.6.2 Gestión de proyectos

Como se ha comentado previamente, para la gestión de archivos de los proyecto existen dos métodos. En primer lugar se puede cargar un proyecto, o crear uno nuevo si no se encuentra el proyecto deseado. Y, en segundo lugar, se puede guardar el proyecto activo, siempre y cuando algún proyecto se encuentre activo.

El guardado de proyectos se realiza desde la propia ventana principal al interactuar con el botón correspondiente. El cargado se realiza desde una ventana dedicada a tal fin. Aquí es necesario indicar el nombre del proyecto que se desee cargar, o comenzar.

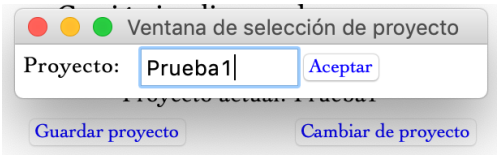

Fig. 23: Ventana de cargado/creación de proyectos

6.6.2 Gestión de recursos

Para cada tipo del recurso del proyecto, trabajadores y tareas, se pueden realizar tres acciones: añadir uno nuevo, listar los existentes y editar el recurso (desde donde se pueden eliminar).

Añadir trabajador

La ventana diseñada para añadir trabajadores contiene campos rellenables para introducir los datos identificativos y de desempeño del trabajador. Esta pantalla se puede observar en la figura 24.

En la parte superior aparecen dos campos, en ellos se introducen el nombre y el apellido (o apellidos) del trabajador. Estos se emplean únicamente de cara al usuario, para distinguir a los trabajadores entre sí. La aplicación usa el identificador con este objetivo, por lo que no se comprueba que el nombre tenga una estructura correcta; puede ser un número o estar vacío (ya que son cadenas de caracteres válidos).

En la parte central de la ventana se encuentran dos menús desplegables donde se pueden escoger los conocimientos y/o destrezas del trabajador. Si el conocimiento o destreza deseado no se encuentra en el menú, se puede introducir manualmente como texto. En este caso se considera como un nuevo conocimiento y, a partir de ese momento, aparece como opción para escoger en los siguientes trabajadores y tareas. Esto pretende minimizar errores producidos por escribir un mismo conocimiento de dos modos distintos en el trabajador y la tarea.

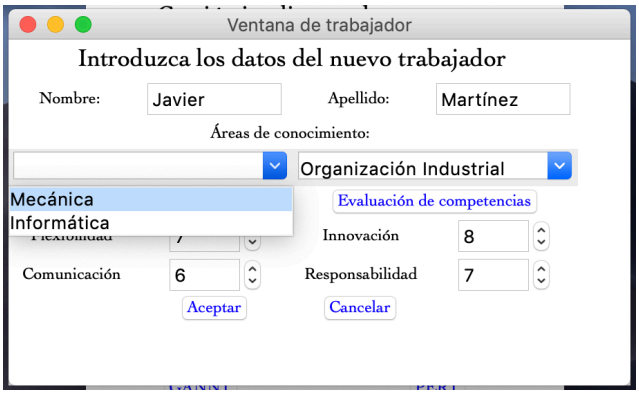

Fig. 24: Ventana de añadir trabajador

En caso de desear añadir más conocimientos o destrezas se debe editar el trabajador desde su apartado correspondiente, donde se permite seleccionar hasta cuatro conocimien-

tos. Para esta demostración no se ha considerado necesario incluir un número mayor, aunque los algoritmos están diseñados para trabajar correctamente independientemente del número de conocimientos.

En la parte inferior de esta pantalla se localizan los campos donde se introduce la valoración de cada competencia del trabajador. Se trata de campos que solo admiten datos numéricos entre 0 y 10. En caso de introducir un dato no numérico, la aplicación no lo procesaría, y esa competencia queda valorada con la puntuación mínima.

Si el usuario desea acceder al método de evaluación, se puede interactuar sobre el botón "Evaluación de competencias". Este abre una nueva ventana donde, tras seleccionar la competencia deseada, genera un formulario de evaluación. Al rellenar este formulario y pulsar sobre el botón "Calcular", se evalúa el formulario, y se devuelve la puntuación correspondiente.

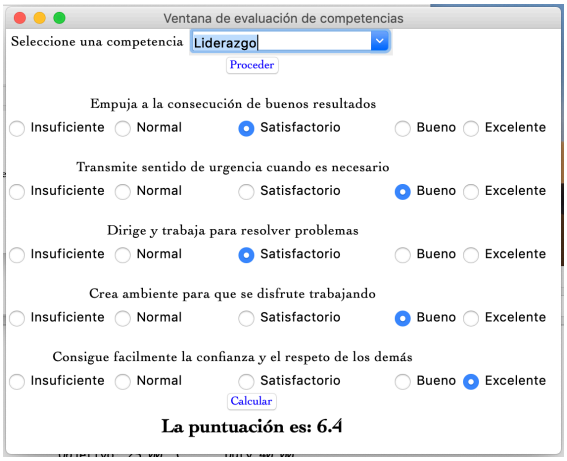

Fig. 25: Ventana de evaluación de competencias

Listar trabajadores

Esta ventana, mostrada en la figura 26, se emplea para visualizar la información de los trabajadores. En ella se genera una tabla con cinco columnas y tantas filas como trabajadores haya en el proyecto. El objetivo es concentrar toda la información manteniendo una estructura clara, donde sea fácil interpretar y buscar información.

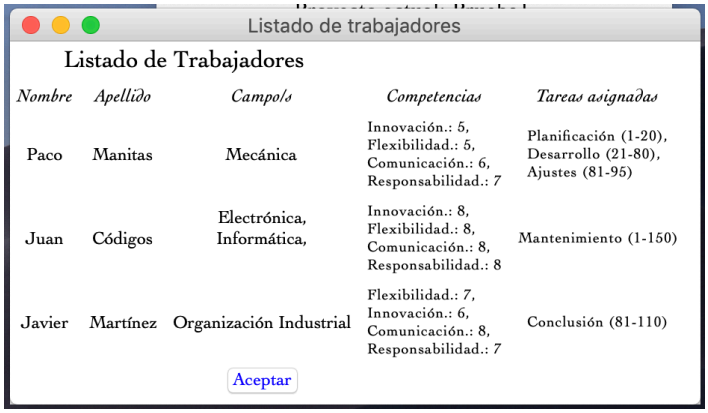

Fig. 26: Ventana de listado de trabajadores

Las cuatro columnas de la izquierda ofrecen la información introducida por el usuario, y que se emplea como datos de entrada por la aplicación: las dos primeras recogen el nombre y el apellido, la tercera presenta los conocimientos y destrezas, y en la cuarta se muestran las competencias y su valoración.

En la última columna aparecen las salidas de la aplicación: las tareas asignadas y las fechas asignadas a cada una. En esta pantalla se puede ver rápidamente el nivel de ocupación de cada trabajador y en qué medida depende el proyecto de su labor.

Modificar trabajador

Esta ventana se emplea para modificar o eliminar trabajadores del proyecto. El primer paso para usarla consiste en escoger el trabajador que se desea editar, o eliminar, mediante un menú desplegable. Después se decide entre modificarlo o eliminarlo con los botones que se encuentran bajo el menú. Al seleccionar la opción de editar el trabajador se genera un formulario similar al empleado para añadir trabajadores.

La principal diferencia es que, en este caso, se dispone de cuatro menús desplegables para seleccionar conocimientos o destrezas.

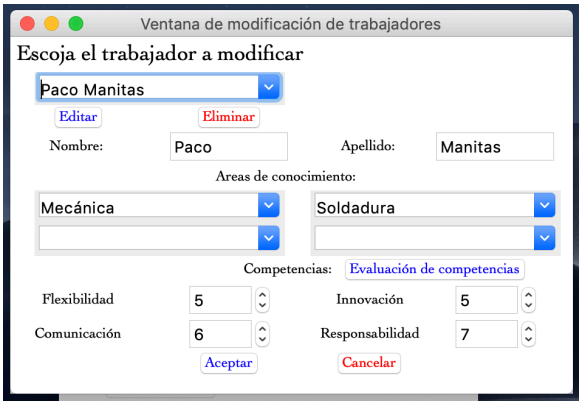

Fig. 27: Ventana de modificación de trabajadores

Añadir tareas

Esta ventana es la que se debe emplear para añadir tareas. El funcionamiento es similar a la ventana de añadir trabajador. En la parte superior se introducen el objetivo y la duración. El objetivo actúa de forma similar al nombre de los trabajadores; su única función es servir de identificador para el usuario.

La duración debe ser un número entero y se verifica antes de asignarlo a la tarea. En caso de tener problemas para la asignación se le da un valor por defecto, que equivale a una duración instantánea. En la parte central se escogen los requisitos de la tarea, al igual que en el caso del trabajador, se pueden añadir nuevos requisitos que aparecerán a partir de ese momento.

La valoración de las competencias en las tareas se realiza de forma distinta a la de los trabajadores. En esta ocasión se emplea una asignación por niveles competenciales, que se deberán determinar de acuerdo a criterios específicos para cada competencia. El usuario debería contar con un manual de gestión por competencias que sirva como guía para poder seleccionar adecuadamente el nivel competencial requerido en cada competencia de cada

tarea. Esto es un paso relevante, ya que la asignación depende en gran medida de estos valores.

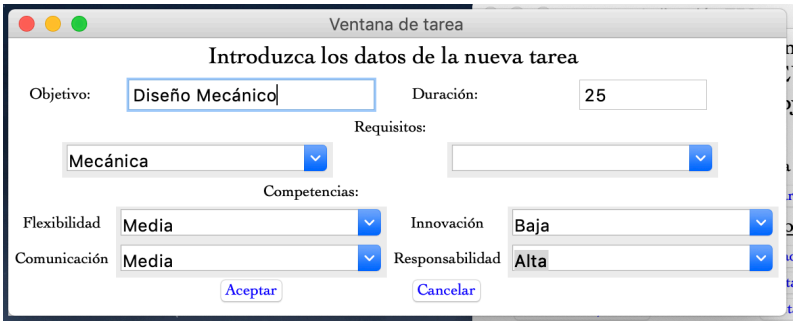

Fig. 28: Ventana de añadir de tareas

Listar tareas

En la ventana de visualización de tareas se debe tener en cuenta que en un proyecto puede incluir una gran cantidad de tareas. Cada tarea contiene más información que los trabajadores, por lo que, resulta importante que toda la información se muestre concentrada y ordenada en una única fila.

En las primeras dos columnas se recogen el objetivo y la duración de la tarea. En las siguientes columnas se presentan los requisitos y los niveles competenciales. En esta ocasión los nombres de las competencias se han abreviado para minimizar la longitud de las filas. Las dos siguientes filas visualizan las dependencias entre tareas, mostrando las tareas previas y las posteriores. La columna final incluye las fechas de inicio y fin de cada tarea; además, aparecen entre paréntesis las fechas de inicio mínimo e inicio máximo. En el caso de las tareas que son parte de la ruta crítica, estas fechas se muestran en negrita para ser más fáciles de identificar. Un ejemplo de la visualización de las tareas se recoge en la figura 29.

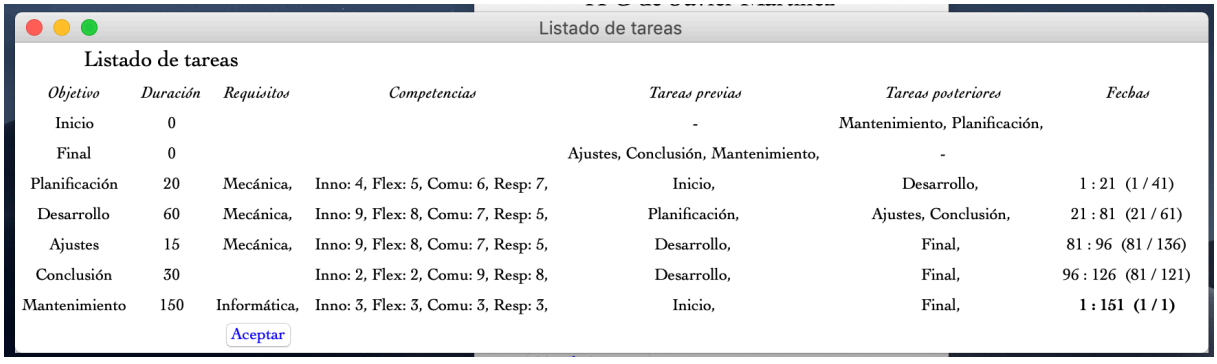

Fig. 29: Ventana de listado de tareas

Modificar tareas

La ventana de modificación de tareas, observable en la figura 30, funciona de forma similar al de la modificación de trabajadores. Tras escoger la tarea, se decide si se desea editarla o eliminarla.

En caso de decidir editarla, se genera un formulario similar al de la ventana de añadir tarea, donde se pueden modificar todos los campos.

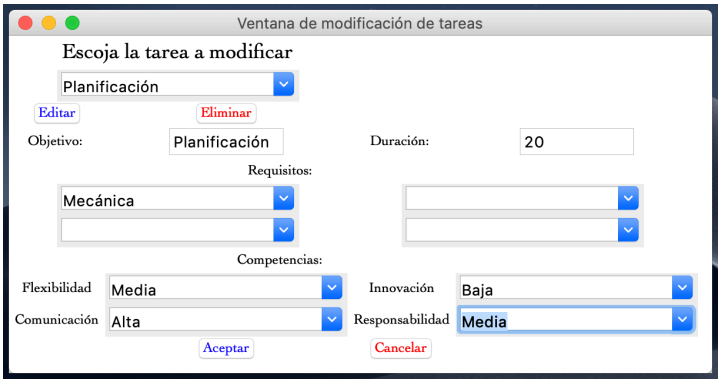

Fig. 30: Ventana de modificación de tareas

6.6.3 Organización del proyecto

Cuando todos los recursos del proyecto se han introducido en el sistema se puede pasar a la organización del mismo, para ello hay tres acciones necesarias que deben realizarse de forma consecutiva: relacionar las tareas, distribuirlas temporalmente y asignar los trabajadores. Solo una de ellas, la relación entre tareas, requiere la interacción del usuario; aun así, se encuentran separadas para poder observar fácilmente los efectos que tiene cada una de estas acciones sobre los trabajadores y las tareas del proyecto.

Relacionar tareas

Las relaciones de dependencia entre las tareas determinan el orden en el que deben realizarse. El usuario debe precisar estas relaciones de forma manual. El método empleado consiste en que, en una ventana nueva, el usuario seleccione una tarea del proyecto y escoja cuántas tareas previas y posteriores requiere. Tras confirmar esta selección, aparecen menús desplegables para escoger cuáles son esas tareas previas y posteriores.

Un ejemplo de este método se muestra en la figura 31.

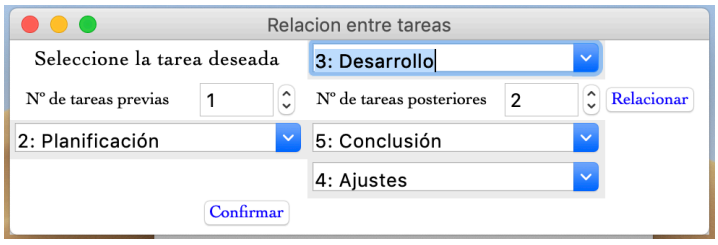

Fig. 31: Ventana de relación de tareas

Cuando se establece una relación de dependencia desde una tarea “A” hacia una tarea “B”, la aplicación genera automáticamente la relación inversa de la tarea “B” a la tarea “A”. Además, cuando se selecciona una tarea para definir sus relaciones, todas las relaciones existentes de esa tarea se muestran. Esto permite modificar fácilmente las relaciones ya existentes.

Organizar tareas

Esta funcionalidad es la encargada de organizar temporalmente las tareas. Las entradas a la misma son la duración de cada tarea y las relaciones de dependencia que se han definido mediante el proceso anterior. El botón correspondiente realiza una llamada al algoritmo

desarrollado en el apartado 6.4.1. De este modo se calculan las fechas de inicio mínimo y máximo de cada tarea del proyecto.

La organización de tareas no genera ninguna ventana ni requiere de ninguna interacción por parte del usuario. Sus efectos se pueden ver al acceder al listado de tareas o a cualquiera de los diagramas. Donde comprobaremos que las tareas ya tienen las fechas asignadas y que el diagrama PERT se representa correctamente.

Distribuir tareas

Para distribuir las tareas del proyecto entre los trabajadores es necesario que estas ya hayan sido organizadas temporalmente, es decir, cuenten con fechas de inicio mínimo y máximo. El botón realiza una llamada al algoritmo de asignación de trabajadores, explicado en el apartado 6.4.2. Al igual que en el caso de la organización de tareas, la función empleada no genera ninguna interacción con el usuario, pero sus efectos pueden comprobarse en el listado de trabajadores y en el diagrama GANNT.

6.6.4 Diagramas

Este apartado detalla las ventanas en las que se representan los diagramas GANNT y PERT. Cada uno de estos gráficos muestra una información distinta, por lo que resulta de gran interés disponer de ambos en los proyectos.

Diagrama GANNT

El objetivo de un diagrama GANNT es mostrar cuándo se realizará cada tarea y cuánto tiempo tiene previsto su desarrollo; aunque en él no se muestran directamente las relaciones entre las tareas. En este trabajo se ha optado por que el diagrama represente a su vez:

- El intervalo de inicios (desde el inicio mínimo hasta el máximo)
- El intervalo de finales (desde el final mínimo hasta el máximo)
- El trabajador encargado de realizar cada tarea.

El intervalo de inicios se representa con un indicador verde en la parte superior de la tarea, mientras que el intervalo de finales se representa con un indicador amarillo en la parte inferior de cada tarea. El trabajador asignado a cada tarea se muestra a su derecha.

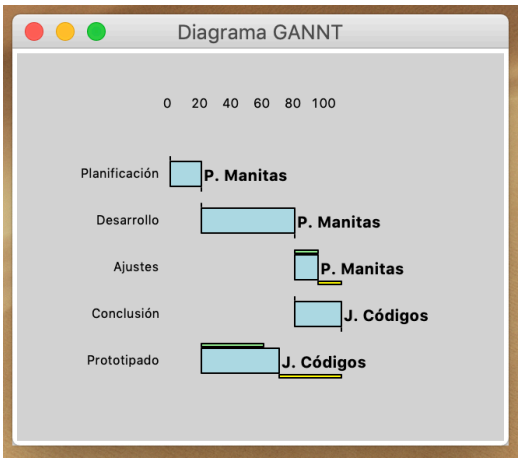

Fig. 32: Ejemplo de diagrama GANNT

El diagrama GANNT es una herramienta muy útil para visualizar de forma rápida y cómoda que tareas generan una mayor carga de trabajo y cuáles disponen de una mayor holgura. También puede servir, aunque de forma indirecta, para ver la carga de trabajo de cada trabajador.

El algoritmo empleado para llevar a cabo este diagrama no requiere de mucha complejidad. Cada tarea se representa en una fila, donde se dibuja un rectángulo desde la fecha de inicio hasta la fecha de fin. Para facilitar la visualización del diagrama, y evitar una disposición confusa de las tareas, cada tarea dibujada se encarga de representar sus tareas posteriores en las siguiente filas mediante una llamada recursiva a la función. Esto organiza las tareas ligeramente sin necesidad de emplear métodos de organización complejos.

Diagrama PERT

El diagrama PERT o malla PERT se emplea como herramienta de planificación y control. Representa las relaciones entre tareas y, aunque de forma menos clara que el diagrama GANNT, la carga temporal de cada una. El algoritmo encargado de transformar la información para poder representar el diagrama PERT se ha explicado previamente en el apartado 6.4.3.

En este diagrama, además de representar la malla, se indican los tiempos e inicio mínimo y máximo, y la ruta crítica. Para mejorar la visualización del diagrama se ha optado por no incluir el nombre las tareas sobre las flechas correspondientes. En su lugar se emplean denominaciones numeradas, con una leyenda en la parte izquierda de la ventana.

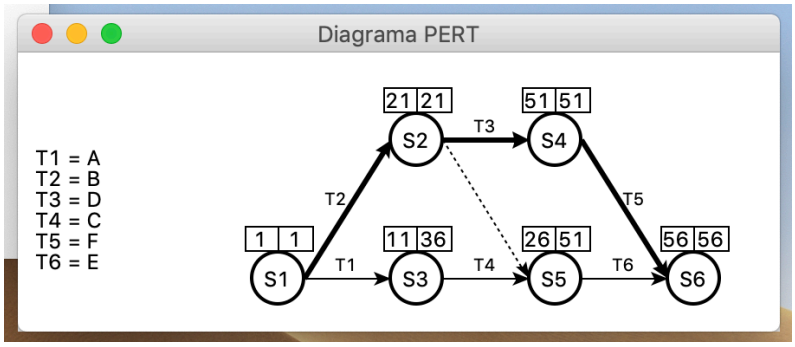

Fig. 33: Ejemplo de diagrama PERT

6.7 Ejemplo

Para demostrar el funcionamiento de la aplicación, a continuación se incluye un ejemplo de la misma. En este se emplean seis tareas y tres trabajadores para demostrar la adecuada gestión y organización de los recursos, así como la distribución temporal de las tareas.

6.7.1 Recursos

Se añaden tres trabajadores al proyecto: el trabajador A no cuenta con conocimientos técnicos, pero sus competencias serán las mejores; los trabajadores B y C tienen los mismos conocimientos, pero se distinguen entre sí en la valoración competencial. El objetivo de estos trabajadores es demostrar las discriminaciones que se realizan para la asignación de cada tarea.

Se determina emplear un total de seis tareas para este ejemplo. Dos de ellas, la inicial (tarea 1) y la final (tarea 4) no requieren estudios. Las otras cuatro tareas sí requieren de un

conocimiento técnico. Estas se distribuyen formando dos ramas en paralelo en el transcurso del proyecto: la rama *a* (tareas 2a y 3a) y la rama *b* (tareas 2b y 3b). De este modo se pretende comprobar la prioridad de asignación que se ha otorgado a la ruta crítica.

Se decide valorar todas las tareas con los mismos niveles competenciales. Por lo que el algoritmo de asignación valorará todas las competencias en la misma medida.

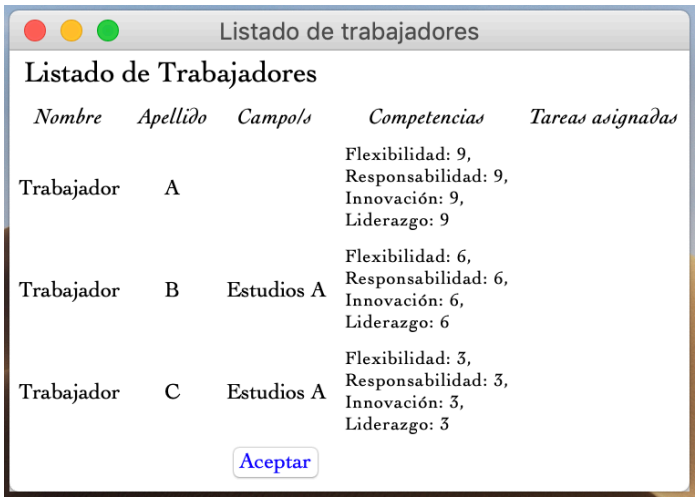

Fig. 34: Listado de trabajadores del ejemplo

Tras fijar las relaciones de dependencia entre tareas y calcular los límites temporales de cada una, se comprueban el listado de tareas y el diagrama PERT del proyecto, mostrados en la figura 35. Se puede comprobar que las tareas no tienen fecha de inicio real válida, -1, ya que no tienen un trabajador asignado. Sí que cuentan con una fecha de inicio mínimo y máximo, recogidos entre paréntesis en la columna derecha.

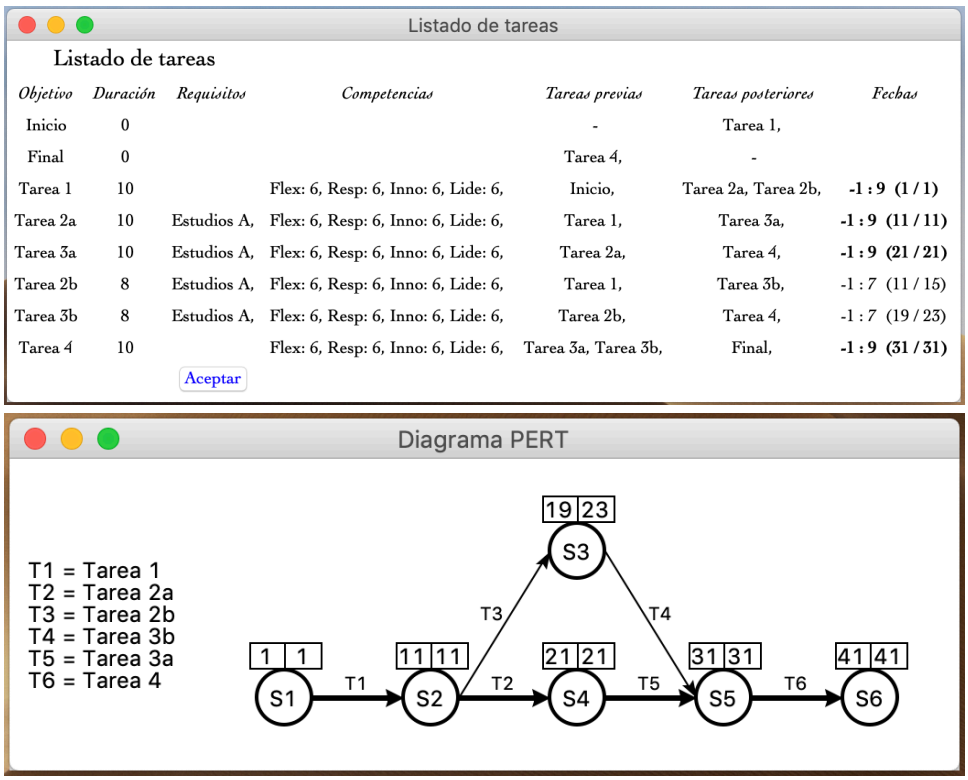

Fig. 35: Listado de tareas y diagrama PERT del ejemplo

Una vez verificado que el árbol de tareas del proyecto es el adecuado, se distribuyen las tareas entre los trabajadores. El resultado de esta operación se puede ver en el listado de trabajadores, figura 36; en el listado de tareas, figura 37; y en el diagrama GANNT, figura 38.

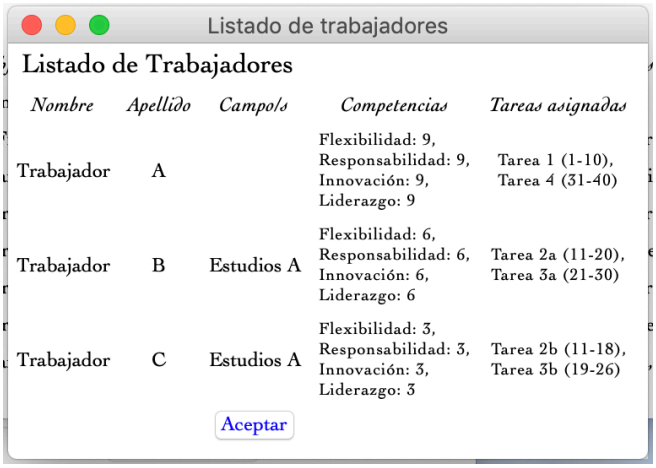

Fig. 36: Listado de trabajadores con tareas asignadas

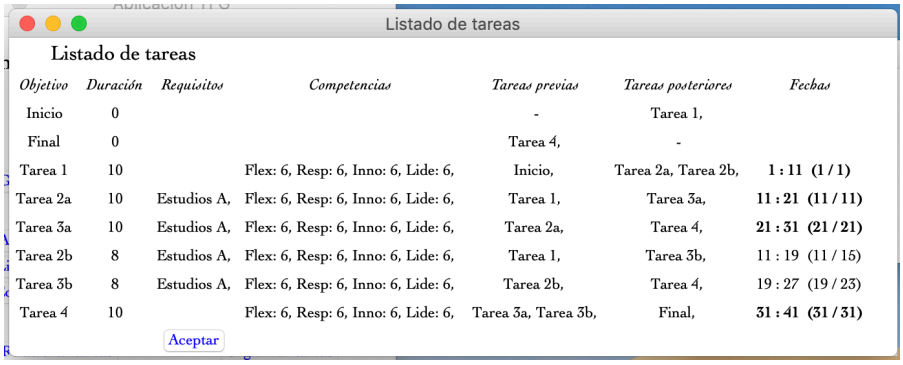

Fig. 37: Listado de tareas con trabajadores asignados

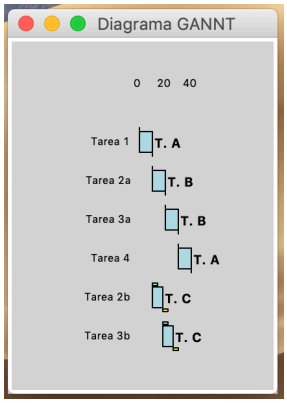

Fig. 38: Diagrama GANNT del ejemplo

En las tres figuras anteriores se puede comprobar que las tareas más exigentes, 1 y 4, se asignan al trabajador más capaz, trabajador A. El resto de las tareas no se le pueden asignar ya requieren de los conocimientos técnicos de los trabajadores B y C. Por lo tanto, se asignan aquellas tareas de la ruta crítica (2a y 3a) al trabajador B, al tener mejores niveles competenciales. Las otras dos tareas las realizará el trabajador C, ya que dispone de los conocimientos necesarios y el trabajador B se encuentra ocupado.

7. Diseño de la implementación final

Mediante la aplicación de demostración del capitulo anterior se ha comprobado que el concepto y los modelos planteados son viables. Sin embargo, dicha aplicación se ha diseñado e implementado para trabajar con un pequeño volumen de información, proyectos independientes entre sí y un único usuario.

Para poder implantar el software en una empresa, se deben llevar a cabo una serie de mejoras que permitan asegurar un funcionamiento satisfactorio. A continuación se comentan las modificaciones que se consideran necesarias para esta implantación a mayor escala. También se realiza un pequeño análisis acerca de las posibles arquitecturas hardware que se pueden emplear, ya que la aplicación tiene que ser accesible desde más de un equipo informático. Por último, se propondrán algunos añadidos que aumentan la funcionalidad de la aplicación.

7.1 Modificaciones de diseño

Seguidamente se van a analizar los principales aspectos que se deben modificar de cara a una implementación a mayor escala y se plantearán soluciones para cada apartado.

7.1.1 Procesamiento de la información

Una de las diferencias más evidentes y relevantes entre la implementación de demostración realizada y un caso real, es el volumen de información que existe en el sistema y que resulta necesaria procesar.

Para que la aplicación pueda trabajar con un mayor volumen de información se pueden aplicar dos tipos de soluciones, obteniendo los mejores resultados al combinarlas. La primera de ellas consiste en acelerar el procesamiento de la información, mientras que la segunda radica en reducir el volumen de información que es necesario procesar.

Mejora del procesamiento de información

Para acelerar el procesamiento de la información es necesario buscar partes o planteamientos del código en la aplicación que puedan estar causando un retraso evitable. Estos puntos suelen ser los que requieren de un mayor coste computacional. En la aplicación de demostración estos retrasos los generan fundamentalmente las búsquedas de información y los algoritmos recursivos.

En la aplicación se almacenaban las tareas y los trabajadores en listas. Sin embargo, la búsqueda en listas de *Python* no es muy eficiente, ya que están diseñadas para ser accesibles por índice. La solución para mejorar la velocidad de búsqueda puede consistir en cambiar esta estructura, pasando a emplear un diccionario donde las claves sean los identificadores de los recursos, y los valores sean los propios recursos. De esta forma el acceso a la información de cada tarea se realiza de forma inmediata, tan solo es necesario conocer su identificador, que debe ser único en el proyecto. El principal problema de esta solución es que implica cambiar gran parte del código, motivo por el que no se ha implementado en la aplicación de demostración.

Otro de los problemas es el uso de algoritmos recursivos. La solución consiste en modificar los algoritmos para minimizar la cantidad de llamadas recursivas y de operaciones redundantes. En la aplicación se han implementado llamadas redundantes a funciones para

asegurar el correcto funcionamiento de los algoritmos, evitando que algún elemento no se procese por error. Debido a la reducida cantidad de información esto no supone ningún problema.

Sin embargo, de cara a una implantación en un ambiente profesional, resulta necesario eliminar estas redundancias y evitar las llamadas innecesarias a funciones y métodos, puesto que pueden incrementar considerablemente el tiempo de cálculo y el uso de memoria. Esto se puede evitar realizando comprobaciones antes de la llamada a la función en vez de en el propio cuerpo de la función.

Reducción del volumen de información

En la aplicación de demostración, toda la información acerca de los trabajadores, las tareas y el proyecto se encuentra cargada en la memoria del sistema. Sin embargo, esta no es una opción viable para una implementación a gran escala, ya que la cantidad de información puede llegar a saturar el sistema.

La solución a este problema consiste en tener en memoria únicamente los recursos necesarios en cada momento. Los recursos se encontrarán en la base de datos y no en la memoria de la aplicación. Esto crea una mayor dependencia entre el modelo y la base de datos, por lo que este debe diseñarse adecuadamente. El diseño propuesto de la base de datos se expone en el apartado 7.3 del presente trabajo.

Un ejemplo de este método de funcionamiento aparece claramente cuando se quieren mostrar una serie de recursos por pantalla. Mediante esta filosofía, el controlador le envía a la vista solo los identificadores de los recursos que se desean mostrar. Posteriormente, la vista genera una petición al modelo para obtener la información deseada de la base de datos. Esto minimiza la cantidad de información en memoria, ya que la vista no almacena la información.

El modelo actúa como puente entre la base de datos y el resto de la aplicación. La cantidad de información que debe manejar la aplicación se reduce, pero aumenta su dependencia respecto al modelo. Otra ventaja de este sistema, es la separación entre la base de datos y los módulos de vista y controlador. Las modificaciones realizadas en la base de datos solo afectan al modelo. Aunque las modificaciones en este último pueden afectar a todos los módulos de la aplicación.

7.1.2 Gestión de múltiples proyectos

En cualquier empresa u organización se desarrollan múltiples proyectos de forma simultánea. Los trabajadores raramente se encuentran asignados a un único proyecto en exclusiva durante toda la duración del mismo. Algunos trabajadores, sobre todo aquellos con una gran especialización, se suelen encargar de una pequeña parte de cada proyecto. Participan en todos los proyectos que requieran de esa clase de trabajo especializado, pero únicamente en una tarea. Este sería el caso, por ejemplo, de los soldadores

Por lo tanto, para la implementación a gran escala, resulta necesario desvincular a los trabajadores de los proyectos. La aplicación debe ser capaz de gestionar múltiples proyectos de forma simultánea, pudiendo asignar al mismo trabajador en tareas de proyectos diferentes, siempre y cuando no se superpongan temporalmente ambas tareas entre sí.

La solución a este problema se basa en modificar la estructura de la base de datos: empleando una única base de datos en toda la organización. Esta situación, al contrario que una base de datos por proyecto, permite disponer de una única tabla de trabajadores, accesible por todos los proyectos. Cada proyecto estará representado por una tabla de la base de datos, estando todos ellos en una tabla general de proyectos. Las competencias se encontrarían organizadas de forma similar a los proyectos; con una tabla por competencia que contenga la información necesaria para su evaluación, y una tabla general de competencias.

Esta nueva distribución, que se detalla más en profundidad en el apartado 7.3, permite compartir trabajadores y competencias entre proyectos. Incluso pueden establecerse relaciones entre tareas de distintos proyectos, posibilitando establecer un sistema de uso de recursos. De este modo, se podría indicar que dos tareas requieren de un mismo recurso, por ejemplo una máquina específica, evitando que se lleven a cabo a la vez, aunque ejecuten trabajadores distintos para proyectos diferentes.

7.1.3 Seguimiento y reorganización del proyecto

En la aplicación de demostración se realiza únicamente una distribución al comienzo del proyecto. En un proyecto real es muy improbable que se pueda cumplir un planteamiento inicial sin sufrir ninguna modificación. Durante el desarrollo de cualquier proyecto pueden aparecer imprevistos que requieren de flexibilidad en la gestión del mismo.

Pueden producirse una gran cantidad de modificaciones en los recursos del proyecto debido a factores de distintos ámbitos. Por ejemplo:

- Posibles modificaciones de las tareas:
    - En la duración: por fallo de previsión o falta de recursos.
    - En los requisitos: por fallo en el análisis previo.
    - En las dependencias: por fallo en la planificación o cambio de objetivos.
    - En las competencias: por fallo en la evaluación competencial.
- Posibles modificaciones de los trabajadores:
    - En los trabajadores disponibles: por bajas o altas en la organización.
    - En los conocimientos o destrezas: por asistencia cursos de formación.
    - En las competencias: por fallos en la evaluación o desarrollo personal.
    - En la disponibilidad: por bajas temporales o viajes

Algunas de estas modificaciones pueden llegar a ser evitables mediante un análisis inicial más exhaustivo, pero otras son absolutamente imprevisibles. El objetivo de esta aplicación es ser capaz de reaccionar y reorganizar los proyectos según vaya surgiendo la necesidad. Sin embargo, surgen dos obstáculos para poder llevar a cabo una correcta reorganización.

En primer lugar es necesario conocer en todo momento el estado actual del proyecto. Es decir, hay que hacer un seguimiento de cada tarea y de la fecha actual. Esto permite saber si las tareas se están realizando de acuerdo a la planificación inicial. En caso de que una tarea se retrase, es necesario recalcular el resto de fechas y, si es preciso, modificar la asignación de trabajadores.

El segundo inconveniente se deriva de la propia reasignación de trabajadores de una tarea en desarrollo, que conlleva dos dificultades. En primer lugar existe un problema de tiempo: un trabajador va a necesitar un periodo de adaptación para hacerse cargo de una tarea que ya estaba en marcha. Este tiempo no será muy importante al comienzo de la tarea, puesto que no habrá mucho trabajo invertido. Sin embargo, según la tarea vaya avanzando resultará necesario un mayor tiempo de adaptación para el nuevo trabajador pueda avanzar al ritmo de su predecesor.

Por lo tanto, a la hora de sustituir al trabajador encargado de una tarea es necesario analizar si la mejora que se va a producir, ya sea en esta tarea o en otro punto del proyecto, compensa los tiempos de adaptación necesarios por parte de todos los trabajadores implicados.

Incluso cuando se considera que sustituir al trabajador resulta conveniente para el proyecto, puede aparecer otro problema. Este se debe a las diferencias en los métodos de trabajo y los puntos de vista de diferentes trabajadores.

Cuando un trabajador se hace cargo de una tarea ya comenzada, debería adaptarse al trabajo que ya se haya realizado, aunque el método o el planteamiento empleados no sean los que él hubiera utilizado. De lo contrario, parte del trabajo desarrollado perderá al volver a hacerse con un planteamiento distinto, provocando un retraso mucho mayor al tiempo de adaptación o incluso problemas con otras tareas relacionadas.

Para minimizar este efecto resulta necesario tener en cuenta las competencias del nuevo trabajador que se va a encargar de la tarea. Las competencias como flexibilidad, adaptación o capacidad de comunicación pueden ayudar a un trabajador a continuar una tarea previamente comenzada.

7.1.4 Gestión manual

En la aplicación de demostración el usuario no tiene control directo sobre el resultado de la organización y distribución de las tareas. Su actuación está limitada a definir los conocimientos y niveles competenciales de cada trabajador. Sin embargo, en determinadas ocasiones, puede resultar interesante o necesario realizar modificaciones manuales.

Se pueden tratar estas modificaciones a modo de restricciones en la organización del proyecto. Se considera adecuado implementar dos clases de restricciones en la aplicación: restricciones de tiempo y restricciones de asignación.

Las restricciones de tiempo en las tareas modificarían los atributos de inicio mínimo e inicio máximo. De esta forma podríamos fijar un intervalo más restrictivo que el calculado en cuanto a la fecha de inicio, o incluso fijarlo en una fecha concreta (cuando el inicio mínimo y el inicio máximo son coincidentes).

En el caso del trabajador se podrían incluir fechas arbitrariamente en la lista de días ocupados con el objetivo de limitar las fechas en las que se le pueden asignar tareas. Los métodos actuales deberían de poder aceptar estas restricciones con modificaciones menores .

Las restricciones de asignación servirían para obligar a la aplicación a realizar una asignación concreta. Al igual que con los atributos actuales, esta restricción debería encontrarse tanto en el trabajador como en la tarea. De forma similar se podría implementar otra restric-

ción con el efecto contrario: evitar la asignación de un trabajador concreto a una tarea determinada.

Estas restricciones se añadirían a los modelos implementando nuevos atributos a las clases de *Worker* y *Duty*. De este modo se podrían activar y desactivar sin afectar a los valores normales calculados por la aplicación.

7.1.5 Acceso a la aplicación

Al implantar este tipo de aplicación en una organización, el acceso a la misma no puede estar restringido a una única persona, no se trata de una solución viable en una organización real. Una cantidad considerable de los miembros de la organización debe ser capaz de acceder al sistema, aunque no todos del mismo modo.

El papel de un operario no puede ser el mismo que el del responsable de recursos humanos o que un jefe de proyectos. Es necesario establecer unas modalidades de acceso en función del puesto y responsabilidades dentro de la organización

Las modalidades de acceso, así como los trabajadores en cada modalidad, se diseñarían según las necesidades de la empresa. Vamos a exponer una posible implantación sencilla con tres modalidades.

Modalidad de RRHH

Una modalidad sería el acceso para el departamento de recursos humanos. Los usuarios en este caso tendrían permiso para gestionar toda la información relacionada con los trabajadores. Estos usuarios deberían asegurarse de que todos los trabajadores mantienen actualizada la información relacionada con el desempeño profesional. Además, deberían realizar actualizaciones periódicas de la evaluación competencial, asegurando que la aplicación trabaja con datos fiables y actualizados.

Al poder acceder a todos los trabajadores de forma simultánea, los responsables de RRHH podrían identificar rápidamente las carencias más importantes de la plantilla en materia de conocimientos, destrezas o competencias. También resultaría interesante que estos usuarios tuvieran acceso para ver los requisitos de las tareas de los proyectos actuales o previstos. Con esta información se puede decidir si son necesarios cursos formativos para la plantilla o buscar nuevas incorporaciones con perfiles complementarios a los actuales.

Modalidad de encargado de proyectos

La siguiente modalidad estaría orientada a la gestión de los proyectos, y por lo tanto debería estar reservada a los responsables de proyectos. La función de estos usuarios sería generar las tareas de los proyectos y definir las relaciones entre ellas. Al igual que en la modalidad de RRHH, resultaría conveniente mantener actualizada la información acerca de las tareas. De esta forma se podrían ir reorganizando el proyecto según fueran apareciendo los problemas.

Los usuarios con esta acreditación tendrían acceso a las restricciones mencionadas en el apartado 7.1.4. Esto les permitiría realizar ajustes sobre la organización planteada por la aplicación.

Modalidad de trabajador

La modalidad de acceso de trabajador estaría disponible para todos los demás usuarios. Cada trabajador debería tener acceso, como mínimo, a las tareas que tiene asignadas y las fechas en las que se tienen que realizar. Además, cada trabajador debería poder indicar cuál es el estado actual de la tarea y si ha aparecido algún requisito o dependencia nuevos.

7.1.6 Valoración de competencias

Las competencias empleadas en la aplicación de demostración, junto a sus niveles y sus métodos de evaluación, son una posible solución, pero no son la única alternativa. Cada organización puede valorar competencias distintas y evaluarlas con métodos diferentes. Por lo tanto, resulta necesario tener el control sobre todos estos parámetros.

La implementación en la aplicación de esta funcionalidad podría ser similar al empleado para añadir recursos al proyecto, mediante un formulario. En un primer campo se recogería el nombre de la competencia; tras este se indicarían los posibles niveles competenciales de las tareas; y, al final, se podía introducir el formulario de evaluación para los trabajadores. En dicho formulario habría que indicar una serie de preguntas, sus posibles respuestas y la valoración correspondiente para cada una.

Al igual que con los recursos de los proyecto, resultaría interesante un método para ver toda la información de las competencias, y otro para poder modificar las competencias ya existentes.

Los métodos empleados para estimar el desempeño de un trabajador en una tarea están diseñados para trabajar de forma correcta independientemente de la cantidad de competencias o su denominación. Por lo que no resultaría necesaria ninguna modificacion al añadir nuevas competencias.

7.2 Funcionalidades adicionales

La implementación a mayor escala abre la posibilidad a desarrollar nuevas funcionalidades en la aplicación. Estas funcionalidades no se han incluido en el software descrito en las páginas anteriores por considerarse funciones no necesarias para la demostración de viabilidad. Sin embargo, se trata de funciones que podrían resultar útiles para la implantación en una organización.

7.2.1 Algoritmo de asignación flexible

En el algoritmo de asignación explicado en el apartado 6.4.2 la asignación es inflexible; es decir, cuando un trabajador se asigna a una tarea ya no se puede modificar. Si aparece una tarea que solo puede realizar un trabajador, pero este ya está ocupado, la tarea se queda sin trabajador, incluso cuando un segundo trabajador podría hacerse cargo de esa primera tarea, liberando al trabajador “más capaz”.

El algoritmo de asignación podría replantearse para que si se da esta situación, sea capaz de modificar asignaciones previas. Evitando, siempre que sea posible, que las tareas se queden sin trabajador.

Un posible método de implementación podría consistir en, al detectar esta situación, imponer una restricción que obligue al trabajador a realizar la “nueva” tarea o le prohíba reali-

zar la “vieja”. Tras esto se volvería a realizar toda la asignación desde el principio, permitiendo tener en cuenta la nueva restricción en todas las asignaciones del proyecto.

Sería necesario implementar ambas soluciones y probarlas en un amplio abanico de situaciones para determinar si se trata métodos eficaces para resolver este problema en un ambiente real o generan problemas de recursión infinita en la búsqueda de trabajadores.

Un ejemplo del posible funcionamiento en alto nivel del nuevo algoritmo se muestra en la figura 39.

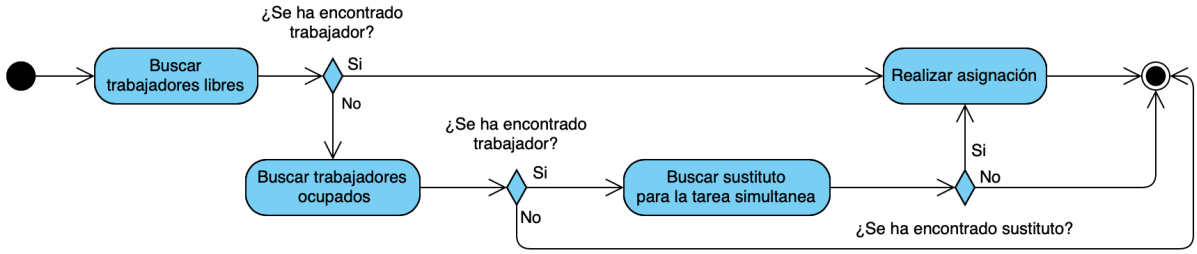

Fig. 39: Posible funcionamiento del nuevo método

7.2.2 Asignación múltiple de trabajadores

La aplicación actual solo tiene la posibilidad de asignar un único trabajador a cada tarea. La solución actual para asignar dos o más trabajadores consiste en duplicar dicha tarea. De esta forma, al haber más de una tarea, el algoritmo intenta asignar un trabajador a cada una.

Puede resultar interesante, en determinadas ocasiones, integrar la opción de escoger el número de trabajadores necesarios para una tarea. Esto simplificaría la visualización de este tipo de situaciones, tanto en el listado de las mismas, como en el diagrama GANNT.

El principal problema aparece debido a los atributos de desempeño. Si todos los trabajadores asignados a la tarea tienen las mismas necesidades no hay ningún problema. Pero si las competencias requeridas no son las mismas, habría que remodelar la clase *Duty* y muchas de las funciones y algoritmos que trabajan con ella, incluido el algoritmo de asignación. Aunque en este caso podría resultar más natural emplear dos tareas distintas con una restricción de que comiencen de forma simultánea.

7.2.3 Coste por trabajador

Uno aspecto de gran importancia en cualquier organización es el apartado económico. Esta aplicación puede calcular rápidamente cuánto tiempo invierte cada trabajador en un proyecto gracias al vector de fechas asignado. Si se introduce en la aplicación el coste económico por hora de un trabajador, resultaría sencillo conocer el coste en salarios que supone un proyecto. Si, además, se incluyeran otros tipos de recursos, como se ha propuesto al final del apartado 7.1.2, también se podrían tener en cuenta para el coste económico.

Este nuevo planteamiento abre la puerta a otra cuestión que puede resultar muy interesante de cara a las empresas. Si se incluyen trabajadores -o empresas- subcontratados en el listado de trabajadores, se podrían conocer el coste del proyecto con trabajadores propios, subcontratados o una combinación de ambos. Esto puede otorgar una gran información de cara a la decisión de qué tareas realizar con trabajadores propios y qué tareas resulta más conveniente externalizar.

También se podrían plantear modificaciones sobre el algoritmo de asignación, para que asigne una puntuación adicional de acuerdo al coste económico del trabajador y la duración de la tarea. Esto permitiría buscar un compromiso entre desempeño y coste, donde la organización podría determinar el peso relativo de cada uno en cada tarea o fase del proyecto.

Sería necesario estudiar más en profundidad los métodos de implementación de estas mejoras, aunque se puede suponer que dependerían en gran medida de los condicionantes de la organización.

7.2.4 Duración de tarea flexible

La última funcionalidad adicional que se plantea en el presente trabajo consiste en alterar ligeramente la duración de las tareas dependiendo del trabajador asignado. En la práctica, se ha considerado como una funcionalidad compleja de implementar y que no aporta necesariamente un gran valor a la aplicación. Aun así, en determinadas situaciones puede resultar interesante, por lo que se va a desarrollar su funcionamiento.

La base de esta funcionalidad es simple: si un trabajador tiene experiencia y competencias más allá de los mínimos requeridos por la tarea, este debería realizar la tarea en menos tiempo del originalmente planteado. De forma análoga, si un trabajador tiene poca experiencia y carece de las competencias adecuadas, tardará un mayor tiempo en completar la tarea.

Con un modelado y algoritmos adaptados a este planteamiento, se podrían ajustar los tiempos del proyecto a los trabajadores desde el comienzo. De lo contrario, estos tiempos dependen de la previsión inicial y de las actualizaciones de estado realizadas de forma periódica.

Valoración de la experiencia

El primer inconveniente que aparece es la valoración de la experiencia. Se considera que las competencias pueden afectar a la velocidad de desarrollo de una tarea, pero la experiencia tiene un papel fundamental. Por lo tanto, en los modelos del trabajador y de la tarea resultaría necesario incorporar métodos para cuantificar esta experiencia. La opción más sencilla sería emplear niveles lingüísticos, de forma similar a los niveles competenciales de las tareas.

La incorporación de estos niveles de experiencia obligaría a una revisión del algoritmo de asignación para tenerlos en cuenta. Sin embargo, habría que estudiar cómo afectarían a la asignación. Podría ser de forma similar al modelo actual, efectuando un filtro acerca de quién puede y quién no puede realizar la tarea; podría implementarse de forma similar a los niveles competenciales, generando una puntuación; o podría emplearse un método mixto.

Efecto de la experiencia

El siguiente paso consistiría en determinar la medida en que se reduce o incrementa temporalmente una tarea debido a las características del trabajador. Se podría aplicar un factor porcentual sobre el tiempo actual teniendo en cuenta la diferencia entre las características del trabajador y las características de la tarea, pero el resultado podría no ser fiable.

Otra solución consistiría en analizar la duración de tareas al ser realizadas por personas con más experiencia y personas con menos, y tratar de obtener una relación entre los parámetros. Es decir, se introduciría cuál es la duración mínima, normal y máxima; y cuáles son

los niveles competenciales en cada caso. Tras esto la aplicación interpolaría la duración estimada

Adquisición de experiencia

Cada vez que un trabajador realiza una tarea adquiere experiencia, por lo tanto, sería necesario modificar el nivel de experiencia de los trabajadores según vayan llevando a cabo actividades en un campo determinado. Esto implicaría analizar en qué grado va a aumentar la experiencia de un trabajador al realizar una determinada tarea. Este incremento resultaría muy complejo de modelar, puesto que depende de una gran cantidad de factores (motivación del trabajador, ambiente de trabajo, duración de la tarea, complejidad, experiencia previa…).

Modificación de tiempos

La implementación de esta funcionalidad genera otro problema en la aplicación. El método actual para gestionar el proyecto es lineal, tal y como se explica en el apartado 6.6.3. La distribución de las tareas requiere que previamente se hayan organizado en la línea temporal. Esta funcionalidad propone que cada asignación de un trabajador modifique la duración de una tarea, afectando a la organización del conjunto. Tras cada asignación de una tarea sería necesario recalcular la organización para todo el proyecto posterior. Lo que sería computacionalmente bastante más costoso.

Sin embargo, esto podría ofrecer una gran ventaja en la práctica, al asignar a los trabajadores óptimos a la ruta crítica del proyecto, dicha ruta se acorta temporalmente. A la vez, el resto de tareas reciben a los trabajadores restantes, por lo que podrían aumentar de duración. Si este proceso se lleva a cabo tras cada asignación, si dos rutas del proyecto tienen duraciones similares, se irían alternando la ruta crítica. Esto provocaría la asignación de los trabajadores óptimos a la rama que en cada momento tenga una mayor duración; y, por lo tanto, se repartieran de forma equilibrada entre ambas.

El principal argumento en contra de este planteamiento está relacionado con el seguimiento del proyecto. Durante este se van modificando las previsiones de duración de cada tarea en marcha; y, por lo tanto, se van recalculando los tiempos. Esto provoca que se realice una continua distribución de las asignaciones. Es decir, la distribución de tareas entre los trabajadores podría cambiar a gran velocidad.

La principal diferencia entre ambos métodos es que el tiempo flexible realiza una estimación temporal de todas las tareas del proyecto en todo momento, aunque requiere de un mayor esfuerzo de modelado e implantación. El seguimiento del proyecto solo permite modificar la duración de las tareas mientras se encuentran en marcha, pero apenas tiene coste de implantación.

7.3 Base de datos

Para realizar una correcta implementación en una organización real resulta de vital importancia la estructura de la base de datos. El diseño de la misma se plasma en el diagrama entidad-relación de la figura 40.

En la parte superior del diagrama se muestran las tablas relacionadas con los proyectos y sus tareas. Cada proyecto tiene un número de identificación único y se divide en tareas. Cada tarea también tiene un identificador único, relacionado con el proyecto, y se divide en subtareas.

Estas subtareas, también con su correspondiente identificador único, son las que se asignan a los trabajadores. Las dependencias entre las tareas se codifican en una tabla independiente, donde se guardarán los identificadores de las dos tareas implicadas en cada dependencia, la previa y la posterior.

La tabla encargada de gestionar los trabajadores no cuenta con la información acerca de sus conocimientos ni destrezas; almacena los datos identificativos y el salario. En su lugar se emplean tablas independientes para ello.

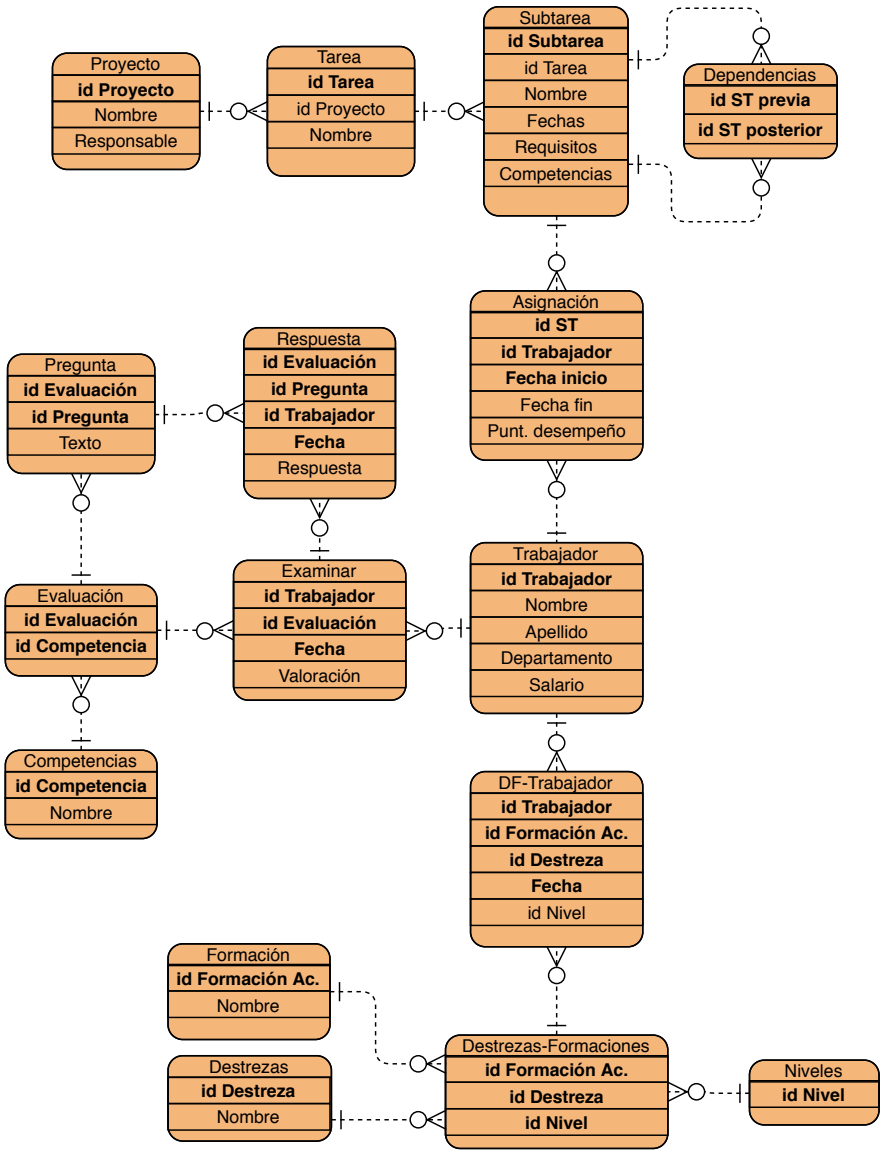

Fig. 40: Diagrama Entidad-Relación de la base de datos

Para gestionar la evaluación de competencias se emplean cinco tablas: listado de competencias, modelos de evaluación, preguntas en cada modelo, respuestas realizadas y evaluación de cada examen. En esta última tabla se relacionan los trabajadores y los exámenes con su valoración. El examen se vincula con la competencia correspondiente en la tabla acerca del modelo de la evaluación.

Para la gestión de destrezas y formación se opta por realizar una modificación respecto al tratamiento de la implementación de demostración. En esta ocasión se relacionan las forma-

ciones y las destrezas entre sí. Es decir, cada formación académica conlleva una serie de destrezas y un determinado nivel. Una tabla adicional se emplea para relacionar cada trabajador con sus destrezas y formaciones académicas. En esta tabla, además, se mantiene un registro de las fechas, para poder evaluar la mejora de los conocimientos de un trabajador a lo largo del tiempo.

7.4 Arquitectura Hardware

Para implementar esta, o cualquier otra aplicación de gestión, en una organización resulta muy importante diseñar correctamente la arquitectura del sistema. Existen tres elementos fundamentales en la aplicación: los usuarios, la lógica y la base de datos. La arquitectura determina qué equipos se encargan de la gestión de cada uno de estos elementos.

Existen varias formas de plantear la arquitectura hardware del sistema, cada una con sus ventajas y sus inconvenientes. A continuación se van a plantear algunas posibles soluciones, junto con sus características principales. Cada empresa debe escoger la opción que mejor se adapta a sus circunstancias.

7.4.1 Arquitectura centralizada

Esta arquitectura se basa en el uso de un servidor central encargado de llevar a cabo todos los procesos. Existen dos opciones en esta misma arquitectura, aunque el funcionamiento y las características son similares. La primera versión cuenta con la base de datos integrada en el servidor principal. La segunda versión separa físicamente los servidores; aun así, el único acceso a la base de datos es a través de peticiones al servidor principal, por lo que de cara al usuario apenas hay diferencia.

En el servidor, o servidores, se encuentran implementados la base de datos, los algoritmos de gestión y los métodos de visualización.

En esta arquitectura, los equipos de los usuarios no efectúan ningún procesamiento de información. Para acceder o modificar la información se realizarían peticiones contra el servidor, que enviaría la información necesaria en cada momento para construir la interfaz de usuario. La aplicación sería una página web, donde el usuario no tiene que realizar ninguna instalación previa a la utilización del producto.

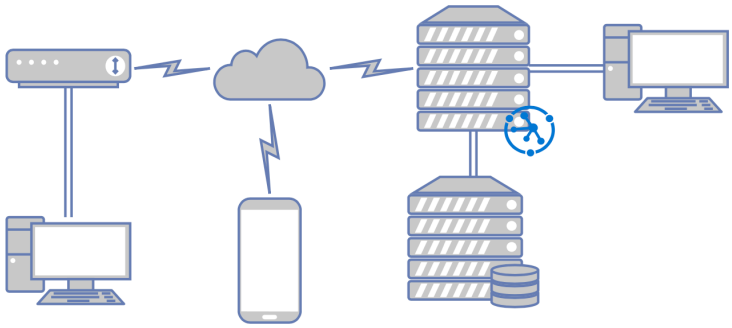

Fig. 41: Diagrama de la arquitectura centralizada con dos servidores

La principal ventaja de este sistema es la comodidad para el usuario final. No requiere de ningún tipo de instalación ni de mantenimiento. El servidor puede estar alojado en una red local, permitiendo el acceso desde todos los equipos de la red, o alojarse en un servidor de

internet, permitiendo el acceso desde cualquier equipo con conexión al mismo. Por lo tanto, se trata de un sistema fácil de expandir, incluso a sistemas portátiles.

Sin embargo, esta arquitectura también cuenta con inconvenientes. El primero de ellos está relacionado con la implantación, donde aparece necesidad de establecer los protocolos de acceso e identificación de usuarios, así como su mantenimiento. Además, en caso de tener la red conectada a internet, la seguridad es un aspecto a tener en cuenta. Ya que toda la información de la empresa podría llegar a ser accesible desde el exterior si no se emplea la seguridad adecuada.

Otro problema se debe a la capacidad de procesamiento del servidor. Si muchos usuarios acceden de forma simultánea, el servidor debe ser capaz de gestionar las peticiones de todos los clientes. Si el número de peticiones de los clientes es muy elevado, el servidor podría llegar a saturarse y dejar de funcionar. Esto implica que la base de datos quede inaccesible para los usuarios.

También durante la actualización del servidor o la base de datos, el sistema quedaría inoperativo. Por lo que se recomendaría disponer de un servidor de emergencia que permitiera sustituir temporalmente al original. Este servidor también debería guardar una copia de seguridad de la base de datos por si hubiera cualquier problema.

7.4.2 Arquitectura descentralizada

Una arquitectura alternativa se basa en emplear un único servidor para la aplicación: el de la base de datos. La implementación de los algoritmos de control se realiza de forma local, en la instalación de cada equipo, evitando la dependencia de un servidor de gestión.

Esta arquitectura tiene dos métodos distintos de implementación. Puede realizarse la implementación completa de la aplicación en todos los equipos o, por el contrario, pueden realizarse implementaciones diferentes en función del usuario que vaya emplear cada equipo. Esta decisión la tomará la organización en función de si prefiere primar la seguridad y el control en la aplicación (instalaciones distintas), o la flexibilidad y facilidad de mantenimiento (una única instalación).

Independientemente del método de implementación, el principal problema de esta arquitectura se debe a los accesos simultáneos a la base de datos. Si dos, o más, usuarios acceden y modifican el mismo campo de la base de datos, pueden surgir problemas. Para evitarlo es necesario implementar algoritmos de control, como los semáforos. Estos secuencian los accesos de escritura, permitiendo la modificación de la base de datos por un único usuario cada vez.

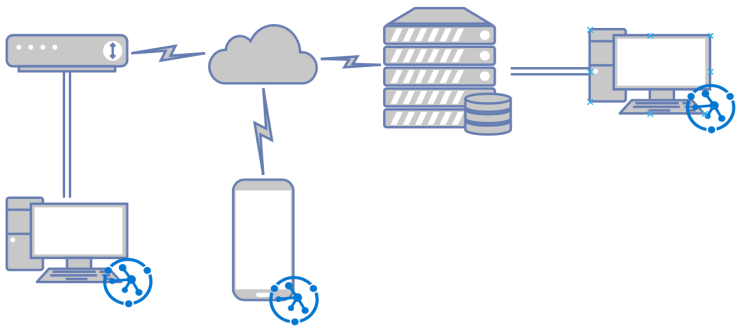

Fig. 42: Diagrama de la arquitectura descentralizada

Por otro lado, la aplicación local en este caso resulta más pesada, puesto que tiene todos los algoritmos de gestión y las funcionalidades deseadas. Esto dificultaría la exportación a sistemas portátiles.

7.4.3 Arquitectura mixta

En este modelo se propone que los equipos de los usuarios tengan acceso de lectura a la base de datos, y que la generación de la interfaz gráfica se realice de forma local.

Al igual que en el primer caso, se emplea un servidor, pero sus tareas quedan limitadas al procesamiento de los algoritmos de gestión de proyectos y la modificación de la base de datos. Además, el servidor de la base de datos y el servidor de los algoritmos serían independientes entre sí.

Con esta arquitectura sigue siendo necesario implementar protocolos de comunicación y de identificación de usuarios. Al menos en los casos en los que el usuario quiera realizar modificaciones sobre la base de datos.

El principal inconveniente de esta distribución es la necesidad de realizar una instalación local en todos los equipos. La aplicación instalada debe permitir, tras la identificación del usuario, el acceso a la información de la base de datos y al servidor de gestión, aunque su nivel puede variar en función del usuario.

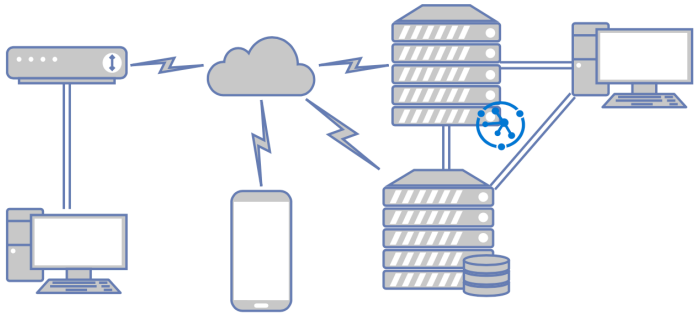

Fig. 43: Diagrama de la arquitectura mixta

Otro problema de esta arquitectura es el mantenimiento de las instalaciones en los equipos. Ya que cualquier modificación o mejora que se desee realizar en la aplicación, tiene que actualizarse d en cada equipo.

Por otro lado, una de las principales ventajas de esta arquitectura es la menor dependencia del servidor, que además soportaría una menor carga de trabajo. Incluso si el servidor se colapsa y deja de responder, toda la información de la base de datos seguiría siendo accesible para los usuarios.

La aplicación local no debe realizar una carga computacional importante, por lo que la exportación a equipos móviles vuelve a ser una opción viable.

8. Conclusiones

La implementación de esta aplicación en una organización real es considerada como una opción posible, siempre y cuando se lleven a cabo las modificaciones planteadas en el apartado 7 del presente trabajo. Es indudable que resultaría necesario invertir una considerable cantidad de tiempo y esfuerzo para asegurar el correcto funcionamiento de la aplicación en todas las posibles situaciones. Ya que no es admisible un fallo del sistema una vez se encuentra implementado y funcionando en una organización.

De acuerdo a las pruebas realizadas sobre el programa de demostración, se considera que una aplicación de este tipo resulta eficaz para optimizar procesos de gestión y organización, así como para reducir la cantidad de trabajo manual necesario. Un software de estas características resulta interesante para cualquier tamaño de organización.

Una empresa pequeña o mediana, que no se suele disponer de los recursos para mantener un fuerte departamento de recursos humanos, podría acceder a métodos de gestión por competencias. Una gran empresa, donde suelen estar implantados sistemas de gestión ERP, podría incorporar esta clase software para ayudar a reducir la carga de trabajo del departamento de recursos humanos. También se puede emplear este software para obtener una visión detallada acerca del estado de los proyectos y la formación o capacidades de los trabajadores.

Se debe tener en consideración que la eficacia y comportamiento de esta aplicación, así como de cualquier otra relacionada con la gestión de personas y proyectos, está estrechamente ligada con el estudio previo al desarrollo. Un análisis de los trabajadores o las tareas distinto, ya sea en enfoque o en profundidad, habría generado una aplicación con un comportamiento diferente.

De forma similar, se concluye que las herramientas y los entornos de desarrollo empleados también influyen en gran medida al resultado final de la aplicación. En este apartado cabe destacar el importante papel que ha tenido en este trabajo el uso de un lenguaje de programación como *Python*. Los lenguajes de programación interpretados permiten un desarrollo a gran velocidad y una alta versatilidad frente a otros lenguajes de programación más convencionales. Aunque también hay que tener en cuenta la importancia de que el lenguaje disponga de una gran cantidad de documentación online, especialmente aquella relacionada con los módulos empleados (*tkinter* y *sqlite3*). La falta de información adecuada puede dificultar en gran medida la implementación, las modificaciones y el mantenimiento de la aplicación.

Se considera que el resultado del trabajo ha sido satisfactorio. Se han puesto en práctica los pasos necesarios para desarrollar un producto software innovador de forma adecuada. Al comenzar por un estudio teórico, y continuar con el desarrollo de un prototipo de demostración se consigue que las posibilidades de éxito de la implementación final sean mucho mayores. En el caso de una aplicación informática resulta relativamente sencillo realizar modificaciones menores, frente a un elemento físico. Aun así, un planteamiento correcto desde el origen facilitará el trabajo y las modificaciones posteriores.

Bibliografia

Tejada Fernández, J. y Ruiz Bueno, C. (2016). *Evaluación de competencias profesionales en Educación Superior: Retos e implicaciones*. Barcelona: Educación XX1.

Alles, M. (2002). *Dirección estratégica de recursos humanos. Gestión por competencias: el diccionario.* Buenos Aires: Editorial Granica.

Grupo Rey Ardid (2017). *Manual de competencias*.

Jalote, P. (2002). *Software Project Management in Practice*. Boston: Pearson Education.

Stevens, P. y Pooley, R. (2000). *Using UML: Software Engineering with objects and components*. Edimburgo: Pearson Education Limited.

Puchol, L. (2007). *Dirección y Gestión de Recursos Humanos*. España: Ediciones Diaz de Santos.

Brassard, G. y Bratley, P. (1990). *Algorítmica: Concepción y Análisis*. París: Masson S.A.

Lutz, M. (1996). *Programming Python*. Sebastopol: O'Reilly & Associates, Inc.

Bertino, E. y Martino, L. (1995). *Sistemas de bases de datos orientadas a objetos*. Delaware: Addison-Wesley Iberoamericana.

Sommerville, I. (1984). *Ingeniería del Software*. Madrid: Pearson Educación S.L.

Alcover de la Hera, C. Mª.; Martínez Íñigo, D.; Rodríguez Mazo, F. y Domínguez Bilbao, R. (2004). *Introducción a la psicología del trabajo*. Madrid: McGraw-Hill.

Allen Weiss, M. (1995). *Estructuras de Datos y algoritmos*. Buenos Aires: Addison-Wesley Iberoamericana.

Piattini, M.; Calvo-Manzano, J.; Cervera, J. y Fernández, L. (1996). *Análisis y diseño detallado de aplicaciones informáticas de gestión*. Madrid: RA-MA Editorial.

Python Software Foundation (2001-2019). *Python 3.8.0 documentation*. Recuperado de: https://docs.python.org/3/

Python Software Foundation (1990-2019). *Python interface to Tcl/Tk*. Recuperado de: https://docs.python.org/2/library/tkinter.html

Like Geeks (2018). *Ejemplos De La GUI De Python*. Recuperado de: https://likegeeks.-com/es/ejemplos-de-la-gui-de-python/

Python Software Foundation (1990-2019). *DB-API 2.0 interface for SQLite databases* Recuperado de: https://docs.python.org/2/library/sqlite3.html

Relación de documentos

(X) Memoria .....67 .....páginas

(_) Anexos .....50 .....páginas

La Almunia, a 26 de Noviembre de 2019

Firmado: Javier Martínez Lahoz