Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Ingeniería de Software

Aplicación web para la clasificación de instancias de problemas de secuenciación con múltiples proyectos

Óscar Pérez Cañón

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Ingeniería de Software

Aplicación web para la clasificación de instancias de problemas de secuenciación con múltiples proyectos

Autor: Óscar Pérez Cañón
Tutores: Valentín Cardeñoso Payo
Marta Posada Calvo

A nadie le importa cómo funciona mientras funcione.
Consejero Hamann, The Matrix

Agradecimientos

Han sido muchas personas las que han ayudado a que este proyecto haya salido adelante. En primer lugar agradecérselo a mis tutores, ya que con sus constantes revisiones y correcciones tanto la aplicación como la memoria ha ido por el camino adecuado.

Me gustaría dar las gracias también a mi familia, por haberme apoyado y acompañado durante toda la carrera. Siempre han querido lo mejor para mí y han sabido darme los mejores consejos en los buenos y no tan buenos momentos.

Por último, agradecer a los amigos que me han acompañado durante la carrera. A esos que conocía desde antes y a los que he tenido la suerte de conocer durante estos años, gracias a ellos he podido llegar a este punto en mi vida.

Resumen

En este Trabajo de Fin de Grado se ha desarrollado una aplicación web que permitirá tener un repositorio de instancias para el problema de secuenciación de múltiples proyectos con recursos limitados. Hasta este momento, no existía ninguna aplicación web que permitiese a los investigadores interaccionar con el repositorio subiendo tanto instancias como soluciones a dichas instancias. El desarrollo de esta aplicación web ha consistido en el análisis, diseño e implementación de una base de datos donde almacenar la información de estas instancias, una API-REST que gestione los accesos a esta información y un servicio web que es quien que permite a los usuarios interaccionar con la base de datos, bien sea para consultar o añadir nueva información. Como resultado del proyecto, se ha obtenido una aplicación web de uso sencillo y seguro, que permitirá compartir la información entre la comunidad de investigadores.

Palabras clave: API-REST, angular, python, aplicación web, RCMPSP

Abstract

In this Final Degree Project we have developed a web application that will allow us to have a repository of instances for the resource constrained multi project scheduling problem. Until now, there was no web application that allowed researchers to interact with the repository by uploading both instances and solutions to these instances. The development of this web application has consisted of the analysis, design and implementation of a database to store the information of these instances, an API-REST to manage access to this information and a web service that allows users to interact with the database, either to query or add new information. As a result of the project, a simple and secure web application has been obtained, which will allow information to be shared among the research community.

Key words: API-REST, angular, python, web application, RCMPSP

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

La gestión de una cartera de proyectos consiste en planificar y programar la ejecución de las actividades que forman parte de cada proyecto de la cartera teniendo en cuenta los recursos limitados que tiene la empresa y los requerimientos específicos para la ejecución de cada actividad expresados en términos de precedencia, buscando optimizar el beneficio para la consecución de la estrategia empresarial [1]. A la optimización de este tipo de problemas se le denomina Problema de Secuenciación de Múltiples Proyectos con Recursos Limitados, RCMPSP por sus siglas en inglés (*Resource Constrained Multi Project Scheduling Problem*).

Por lo tanto, la gestión de múltiples proyectos se convierte en un problema matemático de optimización de una función objetivo sujeto a diferentes restricciones. La primera se refiere a las relaciones de precedencia, las actividades no pueden comenzar a ejecutarse hasta que sus precedentes hayan finalizado. La segunda restricción está relacionada con la utilización de recursos específicos para llevar a cabo cada actividad del proyecto. Estos recursos están limitados y disponibles en cantidades fijas durante el tiempo que dura cada actividad. Teniendo en cuenta ambas restricciones, se deben encontrar los tiempos de inicio de cada actividad de todos los proyectos involucrados (lo que se denomina *scheduling*) de forma que se optimice un valor establecido como objetivo para la gestión de proyectos, por ejemplo, el tiempo total necesario que se necesita para ejecutar todos los proyectos de la cartera (conocido como *makespan*) [2].

Los primeros intentos de resolución datan de mediados del siglo XX y, debido a la alta complejidad matemática y la limitación de las herramientas de optimización, el RCMPSP se dividió en la optimización independiente de cada proyecto de la cartera, siendo conocido como Problema de Secuenciación de un Proyecto con Recursos Limitados, RCPSP (*Resource Constrained Project Scheduling Problem*). Sin embargo, la optimización independiente de cada proyecto no asegura la factibilidad de la ejecución del conjunto de proyectos de forma simultánea, al darse múltiples violaciones en las restricciones del uso de recursos. Por eso, fue necesario el desarrollo de estrategias de optimización más potentes (como *genetic algorithms*, *differential evolutionary*, *particle swarm optimization*, *agent-modeling*, etc.) que permitieron la resolución conjunta de los proyectos de forma que se aseguraba la factibilidad de la solución obtenida.

La comunidad científica que desarrolla dichas estrategias de optimización necesita de un banco de ejemplos (instancias) con los que poder demostrar la capacidad de sus algoritmos de optimización para obtener soluciones válidas desde el punto de vista de cumplimiento de las restricciones, en un tiempo de ejecución computacional aceptable y obteniendo valores de la función objetivo cada vez mejores. Por ello, en primer lugar, es necesario disponer de un repositorio de instancias de acceso público para todos los investigadores. Y en segundo lugar, establecer criterios que clasifiquen las instancias según su dificultad para ser resueltas, pudiéndose dividir entre

fáciles o difíciles, y no sólo por el tamaño (número de proyectos, número de actividades de cada proyecto), o por la topología de la red que se forma con las relaciones de precedencia, sino principalmente por la limitación de los recursos (por ejemplo, fuertemente limitante cuando hay pocas unidades de algún recurso y muchas actividades que deben utilizarlo) o por la distribución temporal de la utilización de recursos a lo largo del tiempo de ejecución de todo el proyecto.

Por otro lado, igualmente importante a tener las instancias con acceso público, es poder acceder a las mejores soluciones encontradas para cada una de las instancias del repositorio. Sin este dato, la comparación entre diferentes algoritmos sería mucho más complejo si es que se pudiese realizar. Pero, las soluciones lo son para una determinada función objetivo. Por lo que, el repositorio de instancias debe permitir almacenar las mejores soluciones según la función objetivo con la que se ha obtenido. Como ya hemos comentado anteriormente, una posible función objetivo es la relacionada con la duración total de la cartera de proyectos, o *makespan*, pero hay otras muchas. Algunas relacionadas con el retraso respecto a una fecha de entrega de cada proyecto pactada con el cliente (en inglés *due date*), o retraso respecto a lo que podría durar cada proyecto si no hubiese limitación de recursos (en inglés *delay*).

Si se realiza una búsqueda en la bibliografía científica sobre el tema, desde el problema sencillo RCPSP pasando por el RCMPSP, se percibe que para el primero si existe un repositorio con instancias ampliamente utilizadas por los investigadores, mientras que para el segundo se han encontrado varios repositorios pero no cumplen los requisitos necesarios como para que puedan ser realmente útiles para los investigadores. Este trabajo de fin de grado pretende rellenar este vacío existente.

1.1 Motivación

El grupo de investigación donde desarrolla su investigación la cotutora Dra. Marta Posada trabaja desde hace años en el desarrollo de estrategias de optimización aplicados al RCMPSP, topándose siempre con el problema de no tener un conjunto de instancias comunes al resto de investigadores para poder hacer la validación de los algoritmos desarrollados. Por lo que empezaron a desarrollar una herramienta para analizar la estructura y dificultad de las instancias, con el objetivo de crear en el futuro un repositorio de instancias. A través de conocidos comunes entre ingenieros industriales, donde desarrolla su labor docente el grupo de investigación al que pertenece la cotutora, e ingenieros informáticos, donde el graduando realiza sus estudios, coincidieron las necesidades del desarrollo de este servicio web y de un tema para el Trabajo de Fin de Grado. De esta colaboración surgió la motivación del desarrollo de este proyecto: la necesidad de tener un servicio web.

Se trata de una aplicación web donde, de manera sencilla, los investigadores puedan, sin necesidad de registrarse, acceder a las instancias disponibles conociendo el valor de los parámetros que las clasifica. Y, por otro lado, los investigadores que deseen subir nuevas instancias, y así poder analizarlas, o subir soluciones de las instancias que ya existen en el repositorio deben poder registrarse. De esta forma, tendrán la posibilidad de proporcionar sus datos de contacto, así como poder actualizar las soluciones que ha subido anteriormente, y proporcionar información sobre la forma en la que se ha obtenido dicha solución como, por ejemplo, la estrategia de optimización, o los parámetros de la experimentación.

Durante el desarrollo del proyecto, las investigadoras han mantenido el rol de clientes del proyecto definiendo las necesidades, características y alcance del servicio web requerido, mientras que el alumno ha asumido el rol de desarrollador. Una vez terminado el proyecto, se realizará la entrega del servicio web pasando a ser las gestoras, manteniéndose el alumno como asesor para futuras ampliaciones.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado es desarrollar una aplicación web que permita la gestión y administración de instancias para el RCMPSP, así como su clasificación. El servicio web funcionará principalmente como repositorio de acceso público de estas instancias, que es uno de las necesidades que tienen los investigadores actualmente.

Adicionalmente, se plantean los siguientes objetivos secundarios:

- Implementar un sistema que permita a los usuarios acceder a los datos a través de un navegador. El servicio estará expuesto en Internet y la interacción con los datos se realizará mediante interfaces sencillas e intuitivas.
- Implementar un entorno que permita gestionar y administrar la información que se ha introducido en el sistema, como los datos de usuarios, las instancias y experimentos.
- Permitir que los usuarios puedan introducir en el servicio sus propias instancias, haciendo que el sistema las almacene y clasifique.
- Permitir que los investigadores puedan subir un *link* al alojamiento de su experimento, permitiendo que otros usuarios puedan utilizarlos para contrastar estadísticamente los nuevos experimentos que proponen.

1.3 Alcance

En este trabajo nos vamos a centrar exclusivamente en el *Front-end* y *Back-end* del sitio web que proporcione acceso al portal de administración de instancias de RCMPSP, tal y como se ha detallado en el apartado de objetivos.

A continuación detallaremos lo que queda fuera del alcance y los motivos que nos ha llevado a su exclusión:

- No se realizará un análisis de las instancias, ya que nos hemos encontrado que en la bibliografía científica hay varios formatos para definirlas y es necesario hacer una aplicación intermedia que sea capaz de traducirlas al formato de entrada de la aplicación ya desarrollada.
- No va a será objetivo de este trabajo de fin de grado, realizar el desarrollo de la gestión de soluciones de cada instancia. Ya que antes de permitir que un investigador suba la instancia y quede registrada, el servicio web deberá comprobar que es una solución válida, es decir que no viole ninguna de las restricciones. Además, se tiene como objetivo permitir visualizar el consumo de recursos a lo largo del tiempo.
- Por último, tampoco vamos a prestar atención detallada a todos los aspectos de seguridad ni de eficiencia computacional o escalabilidad, dado que se trata de construir un prototipo totalmente funcional pero no necesariamente preparado para su puesta en explotación.

Todo ello, será el objetivo de próximos desarrollos.

1.4 Soluciones Existentes

Como se indicó en secciones anteriores, actualmente sólo hay dos servicios web activos en los que se hace un estudio de las instancias:

- https://www.projectmanagement.ugent.be/. Se trata de una web en la que el grupo de investigación *Operations Research and Scheduling (OR&S)* de la Facultad de Economía y Administración de Empresas de la Universidad de Ghent (Bélgica) publica sus actividades. Se dedican no solo a la gestión de proyectos, sino que también a cualquier tema de investigación, abordando el problema desde el punto de vista de la Investigación de Operaciones, junto con los problemas de planificación, como la planificación de proyectos, personal, de máquinas, etc [3].
- https://project.rodrigomartin.dev/. Esta web se dedica a categorizar las instancias que cualquier investigador haya desarrollado y suba. Presenta mucha información sobre la instancia subida permitiendo a los investigadores entender mejor su comportamiento y así poder desarrollar soluciones que permitan optimizar mejor los problemas.

Sin embargo, ninguna de estas páginas ofrece la posibilidad de que los investigadores puedan publicar instancias y que otros puedan acceder a ellas. Las limitaciones existentes en estos sitios web justifican la necesitad de desarrollar un banco de instancias donde puedan ser clasificadas según su dificultad en base a varios parámetros y con las mejores soluciones encontradas según cada función objetivo. Así mismo, es importante añadir una base de datos de algoritmos para así poder compartir la capacidad de optimización de éstos y fomentar la búsqueda de una mejor estrategia de optimización.

Por lo tanto, se puede decir que este proyecto implicará un producto que actualmente no existe en el mercado y que es de gran importancia para la investigación de los RCMPSP.

1.5 Contenidos de la memoria

Esta memoria se organiza en capítulos, que contienen los diferentes apartados en que se desarrolla el trabajo:

- En el Capítulo 1 se incluye una introducción al problema, que muestra la necesidad existente que queremos satisfacer con el desarrollo que proponemos, además de los objetivos, alcance, estado de la situación actual y esta sección.
- En el Capítulo 2 se presenta y discute la metodología aplicada, incluyendo la planificación y análisis de costes del proyecto.
- En el Capítulo 3 se presentan los RCMPSP, una breve introducción a la secuenciación de las actividades en proyectos y el planteamiento y formulación de los problemas multi-proyecto. También se indican las funciones objetivo de éstos problemas y cómo los investigadores buscan un modo de disponer públicamente de la información de sus estudios.
- En el Capítulo 4 se presenta el análisis del sistema. Se indican los actores que interactúan con el sistema, los requisitos que se espera que cumpla. También se muestran los casos de uso que se espera que el sistema sea capaz de ofrecer a los usuarios.

- En el Capítulo 5 se presenta el diseño que se va a seguir cuando se implemente la aplicación web. Se explica el patrón MVC y por qué se ha elegido éste patrón de arquitectura software, junto con una explicación del modelo de capas que se seguirá. A demás, se muestra el diagrama de despliegue, modelo de dominio y diagrama de entidad relación que se ha preparado para la implementación del proyecto.
- En el Capítulo 6 se presenta la implementación que se ha aplicado al proyecto. Primero se explicarán las herramientas de desarrollo utilizadas y después una breve explicación de cómo se han cumplido las necesidades detectadas en la fase de análisis y unido a los modelos desarrollados en la de diseño.
- En el Capítulo 7 se presentan las pruebas realizadas a lo largo de la fase de implementación para asegurar el correcto funcionamiento del sistema.
- Finalmente, en el Capítulo 8 se presentan las conclusiones del trabajo, valoración personal del desarrollo del mismo y algunas propuestas de trabajo futuro.

Adicionalmente, se pueden encontrar dos apéndices al final de esta memoria, un manual de instalación y otro de usuario:

- En el manual de instalación (Apéndice A) se indican los pasos que se necesitan seguir para poder desplegar en un servidor esta aplicación web.
- En el manual de usuario (Apéndice B) se indica la funcionalidad de la que dispone un usuario sin registrar, un investigador y un profesor.

Capítulo 2

Metodología

Debido a la clara definición de los objetivos y el alcance del proyecto desde el principio, hemos optado por seguir un modelo incremental para gestionar la planificación. El enfoque incremental en la gestión de proyectos tiene como objetivo el crecimiento gradual de la funcionalidad. Esto significa que nuestro producto irá evolucionando a medida que se realicen los incrementos, hasta que se completen todas las funcionalidades requeridas y se adapte perfectamente a los requisitos establecidos.

Una forma efectiva de dirigir el proceso iterativo incremental es dar prioridad a los objetivos y requerimientos en función del valor que se ofrece al cliente. Los pasos clave en este proceso implican comenzar con una implementación básica de los requerimientos del sistema y, de manera iterativa, mejorar la secuencia evolutiva de las versiones hasta que el sistema completo esté completamente implementado.

En la planificación de este proyecto, es importante resaltar algunas consideraciones previas que han tenido un impacto significativo en su desarrollo y planificación:

- La idea del proyecto se propuso en diciembre de 2022, pero su ejecución real comenzó en una fecha posterior.
- La planificación detallada del proyecto se inició en febrero de 2023 debido a circunstancias personales, se decidió que no se llevaría a cabo una planificación exhaustiva.
- Los días asignados para trabajar en el proyecto son de lunes a viernes, de 5:00 p.m. a 10:00 p.m., debido a compromisos laborales, y los sábados de 10:00 a.m. a 1:00 p.m., y de 5:00 p.m. a 9:00 p.m.
- La falta de conocimiento o familiaridad con algunas herramientas utilizadas en el proyecto ha implicado un incremento en el tiempo de planificación para algunas iteraciones. Esto se debe a la necesidad de aprender dichas herramientas y al ritmo de desarrollo más lento que esto conlleva.

2.1 Fases y costes

2.1.1 Fases de trabajo y planificación

La primera fase del proceso es la fase de Análisis. Se van a tratar de definir los requisitos y funcionalidades de las que va a estar formado el servicio a desarrollar. En esta fase nos vamos a centrar en obtener toda la información necesaria sobre los objetivos que queremos conseguir con el desarrollo del servicio web. Se considera que tres semanas es el tiempo necesario para lograr de manera satisfactoria un estudio suficientemente completo.

La segunda fase será la definición de las tareas y los incrementos. En esta fase se definen las tareas que se deberán realizar para poder desarrollar el proyecto. También se definen los incrementos y las tareas que lo componen para poder tener prototipos del servicio al final de cada uno. Se considera que una semana es más que suficiente para crear unos incrementos eficientes.

La siguiente fase consistirá en el desarrollo de un sistema de gestión de usuarios y su correspondiente servicio web para el acceso de los usuarios. Este será el primer incremental que consistirá en la creación de una API de gestión de usuarios, la definición de la tabla correspondiente en la Base de datos y las interfaces web correspondientes para su uso. Se estima que la duración de esta fase será de dos semanas.

Después pasaremos a la fase de desarrollo de un sistema de gestión de instancias y su correspondiente servicio web. Este segundo incremental consistirá en ampliar la API creada en la fase anterior, añadiendo la funcionalidad necesaria para poder crear y gestionar las instancias que se quieren analizar, junto con la definición de las tablas correspondientes y sus relaciones en la Base de datos y las interfaces web correspondientes. La duración estimada de esta fase será de tres semanas.

La siguiente fase es el desarrollo de un sistema de gestión de experimentos y su correspondiente servicio web. Este tercer incremental implicará ampliar de nuevo la API añadiendo la funcionalidad necesaria para poder crear y gestionar los experimentos que los usuarios quieran publicar, junto con la definición de la tabla correspondiente y sus relaciones en la Base de datos y las interfaces web correspondientes. La duración estimada de esta fase será de una semana.

Luego se encuentra la fase de desarrollo de un sistema de control de sesiones. Este cuarto incremental implicará añadir al prototipo actual la funcionalidad necesaria para gestionar las sesiones de los usuarios, así como la protección de ciertas operaciones en función de los datos de la sesión del usuario. Se estima que esta fase tenga una duración de dos semanas

Entonces se hará una fase de test del proyecto. Este quinto incremental implicará realizar varias pruebas al servicio para comprobar la robustez y la correcta funcionalidad del prototipo. Para poder comprobar la completa funcionalidad, con las restricciones en las operaciones y la validación de datos se estima que esta fase tenga una semana de duración.

Por último, se realizará la memoria del Trabajo de Fin de Grado. Este sexto y último incremental será dedicado al desarrollo y validación de la memoria del proyecto. Se estima que tenga una duración de tres semanas.

En la figura 2.1 se adjunta el diagrama de Gantt correspondiente al desarrollo de las distintas fases del proyecto.

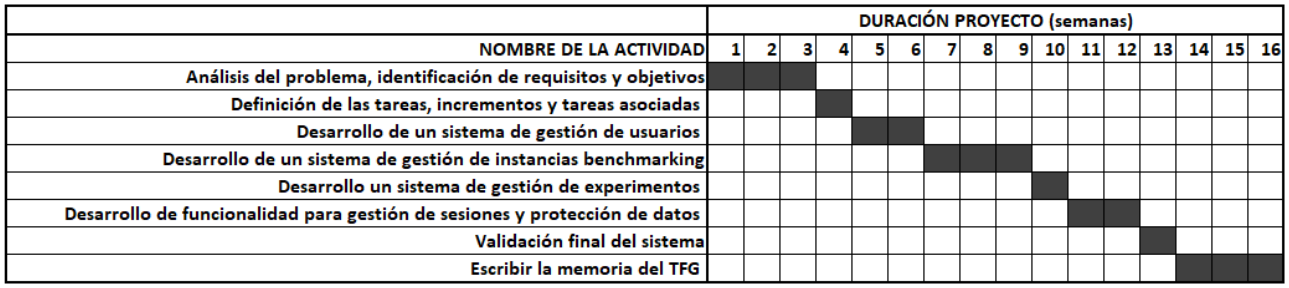

Figura 2.1: Diagrama de Gantt con las fases del proyecto

2.1.2 Costes del proyecto

A continuación, se van a detallar los costes de todas las herramientas, programas y equipo que han sido necesarias para el desarrollo del proyecto.

Costes Hardware

Los costes correspondientes a los equipos tecnológicos se detallan en la tabla 2.1:

Equipos tecnológicos	Precio (€)	Vida útil (años)
Ordenador portátil	800	7
Servidor *cloud*	7 / mes	10
Ratón inalámbrico	25	3
Monitor PC	95	8
Conexión a Internet	50 / mes	1

Tabla 2.1: Costes de componentes hardware

Costes Software

Estos son los costes referentes a las herramientas y programas utilizadas en el desarrollo del proyecto, ver tabla 2.2:

Herramientas/Programas	Precio (€)
Windows 10	0
Office 365	0
Visual Studio Code	0
Python	0
MySQL	0
Angular	0

Tabla 2.2: Costes de componentes software

Al ser una web desarrollada para investigadores de la Universidad de Valladolid, se utilizarán las licencias UVa para Windows 10 y Office 365, por lo que no hay costes asociado a su uso. El resto de herramientas son *open source* y no implican un coste. Por lo tanto, el desarrollo del proyecto tiene un coste único del equipo hardware y del mantenimiento del servidor donde estará alojado el servicio.

Capítulo 3

Marco Conceptual

3.1 Introducción a la secuenciación de actividades en proyectos

El problema de la secuenciación de operaciones es uno de los más estudiados en la gestión de proyectos por su importancia y dificultad. En primer lugar, deberemos dar una definición de lo que se entiende por secuenciación como la determinación del orden en el que deberán ejecutarse el conjunto de actividades que forman el proyecto y su distribución temporal.

Se pueden considerar tres grandes tipos de problemas en la secuenciación:

- Secuenciación de actividades de un único proyecto sin límite de recursos.
- Secuenciación de actividades de un único proyecto con recursos limitados, RCPSP (*resources contraints Project scheduling problem*).
- Secuenciación de actividades de varios proyectos (independientes entre si) que comparten unos recursos limitados. RCMPSP (*resourceds constraints multiproject scheduling problem*).

Se puede decir que esto es una primera clasificación, y que después pueden darse muchos subgrupos, en los que la complejidad del problema va de menor a mayor.

El primer grupo ha sido ampliamente estudiado. Existen herramientas como son el GANTT, CPM, PERT y el PERT-COST. El Gantt fue desarrollado por Henry Laurence Gantt (colaborador de Frederick Winslow Taylor) sobre los años 1910, y muestra de forma gráfica la distribución temporal de las operaciones [4]. Se puede utilizar para problemas con y sin consumo de recursos de tal forma que se visualiza no sólo el inicio y final de cada actividad sino también el consumo acumulado de los recursos con escala temporal. Cuando el número de actividades crece y las relaciones entre ellas se hace más compleja el Gantt es insuficiente y debemos utilizar el PERT o el CPM. El PERT (*Program Evaluation and Review Technique*) fue desarrollado en 1958 por

la Oficina de Proyectos Especiales de la Marina de Guerra del Departamento de Defensa de los EE.UU. [5]. De forma paralela, DuPont desarrolló el CPM (*critical path method*) [6]. Todos estos métodos parten de una secuencia pre-establecida de las operaciones (relaciones de precedencia o restricciones tecnológicas) y de una estimación de las duraciones de cada actividad. Existen herramientas gratuitas que permiten hacer el Gantt, el PERT o el CPM de un proyecto (por ejemplo: http://www.ganttproject.biz/).

El PERT-COST es una modificación del PERT que nos permite gestionar los costes asociados a la ejecución de las actividades para optimizar la duración del proyecto [7].

Cuando se introduce la limitación de recursos aumenta la complejidad y estas herramientas ya no son válidas puesto que no pueden determinar la secuencia más adecuada de las actividades. En este caso, además de las restricciones tecnológicas (relaciones de precedencia) hay que tener en cuenta que las actividades consumen una cantidad de recursos que son comunes a otras actividades. Estos recursos están limitados y por lo tanto habrá que determinar qué actividad tiene prioridad sobre las demás, es decir, determinar la secuencia en la que se realiza el proyecto y se consumen los recursos.

Así, el problema pasa a uno de optimización donde hay una función objetivo, que puede formularse en términos de tiempo o costes, sujeto a unas restricciones que son las de precedencia y las de la cantidad de recursos disponibles.

Los métodos de resolución han pasado por las etapas típicas de cualquier problema de optimización. Los primeros intentos se centraban en la formulación exhaustiva del problema y su resolución mediante técnicas exactas, como programación entera. Después se avanzó por las diferentes técnicas heurísticas, meta-heurísticas y simulación. En los que ya no se busca la solución óptima (imposible de encontrar en los entornos reales debido al número de actividades e imprecisión de los datos), sino al menos una buena solución (la que se obtiene en un tiempo adecuado y que satisface las expectativas del gestor del proyecto).

Aumentando la complejidad del problema, llegamos a la secuenciación de actividades de varios proyectos simultáneos, que consumen recursos comunes. Inicialmente, este problema se resolvió optimizando de forma asilada cada uno de los proyectos, lo que llevaba muchas veces a la mala utilización de los recursos disponibles (por ejemplo, teniendo que multiplicar el número de recursos necesarios). Después se intentó resolver como si se tratara de un solo proyecto introduciendo relaciones de precedencia entre las actividades iniciales y finales de los proyectos (generando una única red de precedencia en las que el nodo inicial y final era lo único común entre proyectos). Pero de esta forma la gestión de todos los proyectos (a nivel conjunto y superior) distaba mucho de ser todavía eficiente. Finalmente, se aborda el problema desde un punto de vista multi-proyecto en el que se optimiza una única función objetivo sujeto a las restricciones de precedencia de cada proyecto tratadas de forma aislada y las limitaciones de recursos conjunto para todos los proyectos. Las técnicas de resolución son las mismas pero adaptándose al entorno multi-proyecto.

3.2 Formulación del problema multi-proyecto

La definición de proyecto según ISO-8402, ISO (1990) (Organización Internacional de Normalización [ISO], 1990): "*Project is a Unique process, consisting of a set of coordinated and controlled activities with start and finish dates, undertaken to achieve an objective conforming to specific requirements including constraints of time, cost and resources.*"

Ampliándose en ISO-1006, ISO (1997) (Organización Internacional de Normalización [ISO], 1997): "*The organization is temporary and only constructed for the projects' life. A project may result in the creation of one or several units of a product or service. The project could involve complex interrelations among activities.*"

Por lo tanto, tenemos que realizar la secuenciación de las actividades que son necesarias para la realización de un proyecto. Para ello partimos de un conjunto de actividades a realizar, un conjunto de recursos con los que podremos realizar las actividades, unas restricciones de precedencia entre dichas actividades y una función objetivo que determinará la bondad de una solución.

La nomenclatura para el problema de secuenciación de un único proyecto es la siguiente. Tenemos A_i actividades a realizar con *i* desde 1 hasta *N*, la duración de cada una de ellas estará dada por d_i. Cada actividad consume r_{ik} recursos del tipo *k*, hay disponible R_k cantidad del recurso *k*, y hay disponibles *K* tipos diferentes de recursos. La relación de precedencia entre las actividades se puede expresar mediante una matriz triangular donde x_{ji} = 1 si la actividad *i* es la precedente directa de *j* y ésta necesita que la *i* esté completamente terminada para que pueda comenzar su ejecución. O x_{ji} = 0 si *i* no es precedente de *j*.

El problema queda determinado mediante una matriz tipo DSM (*Design Structure Matrix*)[1] desarrollado por Steward [8]. Ver un ejemplo genérico en la tabla 3.1 , y uno específico en la tabla 3.2 y figura 3.1. Los elementos de la diagonal de la matriz indican las duraciones de las actividades, la parte superior son 0, mientras que la inferior indica las precedencias. Las columnas de la derecha muestran las cantidades de recursos que utiliza cada actividad.

	A_1	A_2	…	A_i	…	A_N	R_1	…	R_K
A_1	d_1	0	…	0	…	0	r_{11}	…	r_{1K}
A_2	x_{21}	d_2	…	0	…	0	r_{21}	…	r_{2K}
⋮	⋮	⋮	…	⋮	…	⋮	⋮	…	⋮
A_j	A_{j1}	A_{j2}	…	A_{ji}	…	0	R_{j1}	…	R_{jK}
⋮	⋮	⋮	…	⋮	…	⋮	⋮	…	⋮
A_N	A_{N1}	A_{N2}	…	A_{Ni}	…	d_N	R_{N1}	…	R_{NK}

Tabla 3.1: DSM genérico para un proyecto con N actividades y K recursos

Project 1						
	A_1	A_2	A_3	A_4	A_5	Resources
A_1	3					R_1=5
A_2	1	2				R_2=5
A_3			3			R_3=5
A_4	1			4		R_4=5
A_5			1	1	5	R_5=5

Tabla 3.2: DSM ejemplo de un proyecto con 5 actividades

[1] Mas detalles sobre DSM en http://www.dsmweb.org/

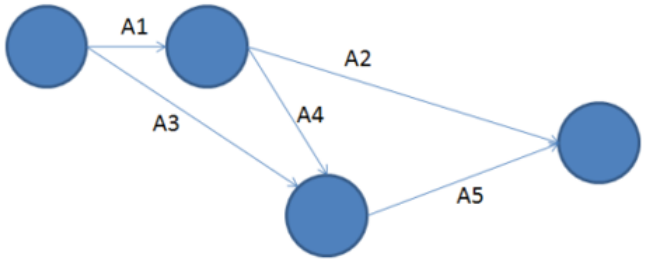

Figura 3.1: Diagrama PERT correspondiente a la tabla 4.2

La función objetivo clásica es minimizar el tiempo de finalización de la última actividad del proyecto (*minimizing the completion time of the project*). Es decir, minimizar C_{max} o el *makespan*. Pero otros son igualmente atractivos como minimizar el retraso o los costes (ver [9] o [10] entre otros).

En cuanto a las duraciones de las actividades, éstas pueden ser interrumpidas (*preemptive*), es decir, que se puede hacer una parte de la actividad y después dejarla en espera hasta que se retoma para su finalización. O por el contrario no es posible interrumpir su ejecución y una vez empieza debe terminar (*non-preemtive*). Además, las duraciones pueden ser valores conocidos (deterministas) o valores con variabilidad (estocásticos).

Los recursos, pueden ser renovables, es decir, que su consumo no las agota. Por ejemplo, la mano de obra, si tenemos 5 trabajadores que son necesarios para una actividad, una vez ésta termina, los 5 trabajadores están disponibles para otra actividad. Los recursos no renovables son aquellos que su consumo los agotan, por ejemplo el dinero (ojalá fuera renovable).

Por último, las actividades pueden realizarse de una única forma (modo) entonces son problemas uni-modales, o de varios modos diferentes, multi-modal. Por ejemplo, una actividad puede precisar una máquina tipo A (problema uni-modal), o puede realizarse en diferentes tipos de máquinas, cada una de ellas con un tiempo de procesamiento y consumo de recursos diferente (problema multi-modal).

Para terminar, cuando tenemos en lugar de un único proyecto a realizar, varios proyectos para su ejecución paralela, estamos ante el problema más complejo multi-proyecto: *RCMPSP* (*resource constraint multi-project scheduling problem*). Los datos de un problema multi-proyecto estarán dados con un DSM para cada proyecto más las restricciones de tiempos de inicio, recursos locales y globales, etc, como podemos ver en la figura 3.2 donde se muestra un ejemplo para tres proyectos. Cuyos recursos son comunes a todos los proyectos y con cantidades de R_1 y R_2 de 6 y 7 respectivamente.

Project 1

	A_{11}	A_{21}	A_{31}	A_{41}	A_{51}	R
A_{11}	3					R_1=5
A_{21}	1	2				R_2=2
A_{31}			3			R_2=5
A_{41}	1			4		R_2=2
A_{51}			1	1	5	R_1=3

Project 2

	A_{12}	A_{22}	A_{32}	A_{42}	
A_{12}	3				R_1=2
A_{22}		1			R_2=5
A_{32}	1		3		R_2=2
A_{42}		1	1	4	R_1=1

Project 3

	A_{13}	A_{23}	A_{33}	A_{43}	
A_{13}	3				R_1=5
A_{23}	1	3			R_2=3
A_{33}	1		4		R_1=3
A_{43}			1	6	R_2=2

Figura 3.2: DSM para un RCMPSP con 3 proyectos

En el RCMPSP hay una serie de características propias que da lugar a nuevas clasificaciones:

- En primer lugar, el momento en el que está disponible cada proyecto pueden ser idéntico (todos los proyectos pueden comenzar en un tiempo relativo cero), o cada uno puede estar disponible en diferentes momentos, en las que las actividades iniciales de cada proyecto tienen su propia fecha de inicio.
- En cuanto a los recursos, pueden ser comunes a todos los proyectos, y las actividades de cada proyecto competirán entre sí para su consumo. O en cambio, cada proyecto tiene sus propios recursos (recursos locales), existiendo o no, un número de recursos comunes a todos los proyectos (recursos globales).

La función objetivo en problemas multi-proyecto se formula también en forma de tiempo de finalización (minimizando el tiempo de finalización del proyecto que finaliza mas tarde), en términos de retraso (minimizando el retraso del proyecto que se retrasa mas, o una media de retrasos) o en función de costes. Para explicar algunas de las funciones objetivas más utilizadas en multi-proyectos se muestra en la figura 3.3 un diagrama Gantt con las fechas de finalización 3 proyectos, siendo CP_j la duración del proyecto *j* suponiendo que no existe limitación de recursos, es decir, la duración del camino crítico (*critical path of the jth Project* CP_j) y D_{Rj} la duración del proyecto *j* cuando si se tiene en cuenta la limitación de recursos (*completion time,* D_{Rj}).

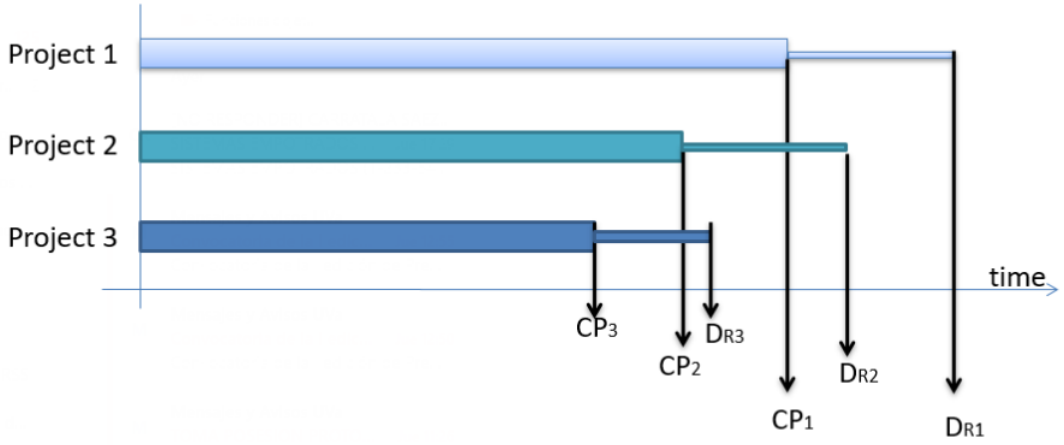

Figura 3.3: Tiempo de finalización con y sin limitaciones de recursos

- Función objetivo respecto del tiempo de terminación: minimizar el tiempo total de finalización (*minimize total makespan*), [11] lo analizó para el problema RCPSP como una generalización del *job shop scheduling problem* (JSP) donde es ampliamente utilizado [9], [12].

$f_1 \rightarrow$ TMS = Max D_{Rj} $\forall$ j = 1 a M donde M es el número de proyectos

En el ejemplo de la figura 3.3 la función objetivo es TMS=D_{R1}

- Función objetivo respecto al retraso total: minimizar la media de la suma de las diferencias entre la finalización de los proyectos con y sin restricciones (*minimize average project delay*), APD. Utilizada en [13], [14], [15], [16].

$$f_2 \rightarrow \text{APD} = \text{MIN } \frac{1}{M}\left(\sum_{j=1}^{M}(D_{Rj} - CP_j)\right) \text{ donde M es el número de proyectos}$$

- Función objetivo respecto al porcentaje medio de retraso (*minimize average percent delay*), analizado en [17]:

$$f_3 \rightarrow \text{MIN}\left\{\frac{1}{M} \times \sum_{j=1}^{M}\left(\frac{D_{Rj} - CP_j}{CP_j}\right)\right\}$$

- Función objetivo respecto al porcentaje de retraso (*minimize percent delay*) analizado en [17]:

$$f_4 \rightarrow \frac{\text{máx}_{j=1}^{M} D_{Rj} - \text{máx}_{j=1}^{M} CP_j}{\text{máx}_{j=1}^{M} CP_j}$$

En nuestro caso nos centraremos en el problema de secuenciación en multi-proyecto con recursos limitados, comunes a todos los proyectos, con misma fecha de disponibilidad de todos los proyectos (tiempo relativo igual a cero), unimodal y con duraciones deterministas, (un ejemplo se muestra en la figura 3.2).

Para los investigadores es muy importante tener un conjunto de problemas que sean de uso común en la comunidad investigadora para poder validar los resultados obtenidos con los diferentes desarrollos. Desde los primeros intentos en la secuenciación, se ha trabajado para poder disponer de librerías en las que obtener los datos y contrastar los resultados. Se puede destacar las siguientes librerías:

- Para el problema de secuenciación (*flow-shop, job shop* y para muchos otros problemas de *Operations Research (OR) problems*) la página http://people.brunel.ac.uk/~mastjjb/jeb/info.html, creada por Beasley en 1990.
- Para problemas del tipo RCPSP tenemos la librería creada por Kolisch: http://129.187.106.231/psplib/ [18], [19], y también la llamada *ProGen/max* [20], http://www.wiwi.tu-clausthal.de/en/abteilungen/produktion/forschung/schwerpunkte/project-generator/rcpspmax/.
- Para el problema RCMPSP se ha creado un generador aleatorio de problemas: http://sbufaculty.tcu.edu/tbrowning/RCMPSPinstances.htm por [17]. Todos ellos de 3 proyectos con 20 actividades cada uno de ellos. La única desventaja que nosotros vemos en esta librería es que no se dispone de datos sobre las mejores soluciones de cada ejemplo (como sucede en el resto de librerías indicadas) ya que los problemas se generan de forma aleatoria.
- Para terminar, destacamos la librería para el problema DRCMPSP [21] con recursos renovables locales mas recursos globales comunes a todos los proyectos (*local renewable resources and a finite set G of global renewable resources shared by all projects.*) http://www.mpsplib.com/.

Desde 2010 en la página www.eis.uva.es/elena están disponibles varios ejemplos tomados de la librería *psplib* y transformándolos en problemas RCMPSP (uniendo proyectos y manteniendo constante la cantidad de recursos). Considerando las llegadas de todos los proyectos idéntica e igual a cero, y sólo con recursos comunes a todos los proyectos. También, se incluyeron varios ejemplos del generador aleatorio de problemas RCMPSP

de Browning y Yassine. De todos ellos, al no ser problemas específicos del RCMPSP no conocemos los óptimos. Por último, se incluyeron cinco ejemplos (MP-MD1, MP-MD2, MP-MD3, MP-MD4 y MP-MD5) de los que si conocemos los óptimos y tienen un tamaño de 10 proyectos con 10 actividades cada uno de ellos, y con 5 recursos compartidos.

Recientemente, más que generar nuevas instancias del problema general RCMPSP, el interés se ha centrado en determinar cómo son estas instancias. Categorizarlas en función de parámetros que midan la dificultad, saturación de recursos, balance de necesidad de recursos a lo largo del tiempo del proyecto. Por ejemplo, en https://www.projectmanagement.ugent.be/research/project_scheduling/RCMPSP [22].

La página web que se desarrolla en este trabajo fin de grado quiere ser banco de instancias, en la que los investigadores puedan subir las instancias creadas por ellos, así como, descargarse las que necesiten. Otro objetivo y el principal, es proporcionar una herramienta a los investigadores para categorizar cualquier instancia en función de diferentes parámetros. Este conocimiento podrá utilizarse para analizar el comportamiento de los diferentes algoritmos de optimización, sus ventajas y sus inconvenientes.

Capítulo 4

Análisis

4.1 Actores del sistema

Un actor es una entidad externa al sistema, que tiene interés en interactuar con él. En este proyecto, se pueden distinguir tres diferentes actores:

- Usuario: Es una persona que acceda e interactúe con el sistema.
- Investigador: Es una persona que usa la página en sus investigaciones.
- Profesor: Es una persona que se encarga de la gestión del sistema.

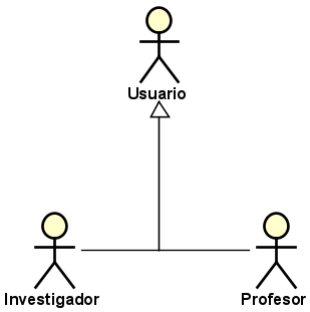

Figura 4.1: Tipos de actores del sistema

4.2 Requisitos

Tras el análisis de los objetivos y el alcance, se han podido obtener los siguientes requisitos que deberá cumplir el sistema.

4.2.1 Requisitos funcionales

Los requisitos funcionales identificados, es decir, la funcionalidad y comportamiento del sistema, están indicados en la tabla 4.1.

Nº	**Descripción**
RF-01	El sistema deberá permitir a los usuarios identificarse en el sistema
RF-02	El sistema deberá permitir a los usuarios registrarse en el sistema
RF-03	El sistema deberá permitir a los usuarios acceder a la lista de las instancias guardadas
RF-04	El sistema deberá permitir a un profesor eliminar una instancia del sistema
RF-05	El sistema deberá permitir a los usuarios subir instancias de un problema al sistema
RF-06	El sistema deberá permitir a los usuarios acceder a una lista de los problemas junto con la mejor solución de cada uno
RF-07	El sistema deberá permitir a los usuarios filtrar las soluciones que se muestren en función parámetro que se quiere optimizar
RF-08	El sistema deberá permitir a los usuarios acceder a los datos de una instancia
RF-09	El sistema deberá permitir a los usuarios acceder a la lista de soluciones propuestas para una instancia
RF-10	El sistema deberá permitir a los usuarios filtrar la lista de soluciones de una instancia en función del método seleccionado
RF-11	El sistema deberá permitir a los usuarios acceder a la lista de experimentos publicados en el sistema
RF-12	El sistema deberá permitir a un profesor eliminar un experimento del sistema
RF-13	El sistema deberá permitir a un usuario subir un nuevo experimento al sistema
RF-14	El sistema deberá permitir a un usuario modificar su dirección de página web
RF-15	El sistema deberá permitir a los usuarios acceder a los datos de los profesores encargados de la gestión del servicio, así como a los datos de los implicados en su desarrollo
RF-16	El sistema deberá permitir a los usuarios acceder al perfil de otro usuario
RF-17	El sistema deberá permitir a un usuario modificar los datos de su perfil

Tabla 4.1: Lista de requisitos funcionales

4.2.2 Requisitos no funcionales

Los requisitos no funcionales, es decir, los que especifican criterios que pueden usarse para describir las características de funcionamiento de un sistema, están indicados en la tabla 4.2.

Nº	**Descripción**
RNF-01	El sistema deberá utilizar MySQL como sistema gestor de base de datos.
RNF-02	El sistema deberá responder de modo fiable las 24 horas del día, todos los días de la semana, exceptuando durante las actualizaciones de seguridad programadas por el administrador del sistema.
RNF-03	El sistema deberá permitir a un usuario aprender a utilizar todas las funcionalidades del sistema en menos de media hora.
RNF-04	El servicio deberá correr sobre un sistema GNU/Linux.
RNF-05	El servicio responderá en un tiempo inferior a 3 segundos.
RNF-06	El servicio será compatible con cualquier ordenador que disponga de un navegador web compatible con HTML5 y Javascript.
RNF-07	El sistema deberá ser capaz de soportar el acceso concurrente de varios usuarios.

Tabla 4.2: Lista de requisitos no funcionales

4.2.3 Requisitos de información

Los requisitos de información, es decir, los que describen los datos con los que el sistema opera, están indicados en la tabla 4.3.

Nº	**Descripción**
RI-01	El sistema deberá tener registrados de un usuario su nombre, apellidos, *email*, contraseña, rol, fecha de registro y el enlace a su página web
RI-02	El sistema deberá tener registrados de una instancia su nombre, el usuario que la creó, las unidades de cada recurso global que consume y una lista con cada problema que ejecuta y el tiempo de inicio de cada uno
RI-03	El sistema deberá tener registrados de un experimento su nombre, el usuario que lo creó, un comentario, la fecha de creación y el enlace a su DOI

Tabla 4.3: Lista de requisitos de Información

4.3 Casos de uso

Una vez obtenidos los requisitos del sistema, se pueden visualizar de una manera más clara utilizando el diagrama de casos de uso en la figura 4.2 para ver las interacciones entre el sistema y el usuario.

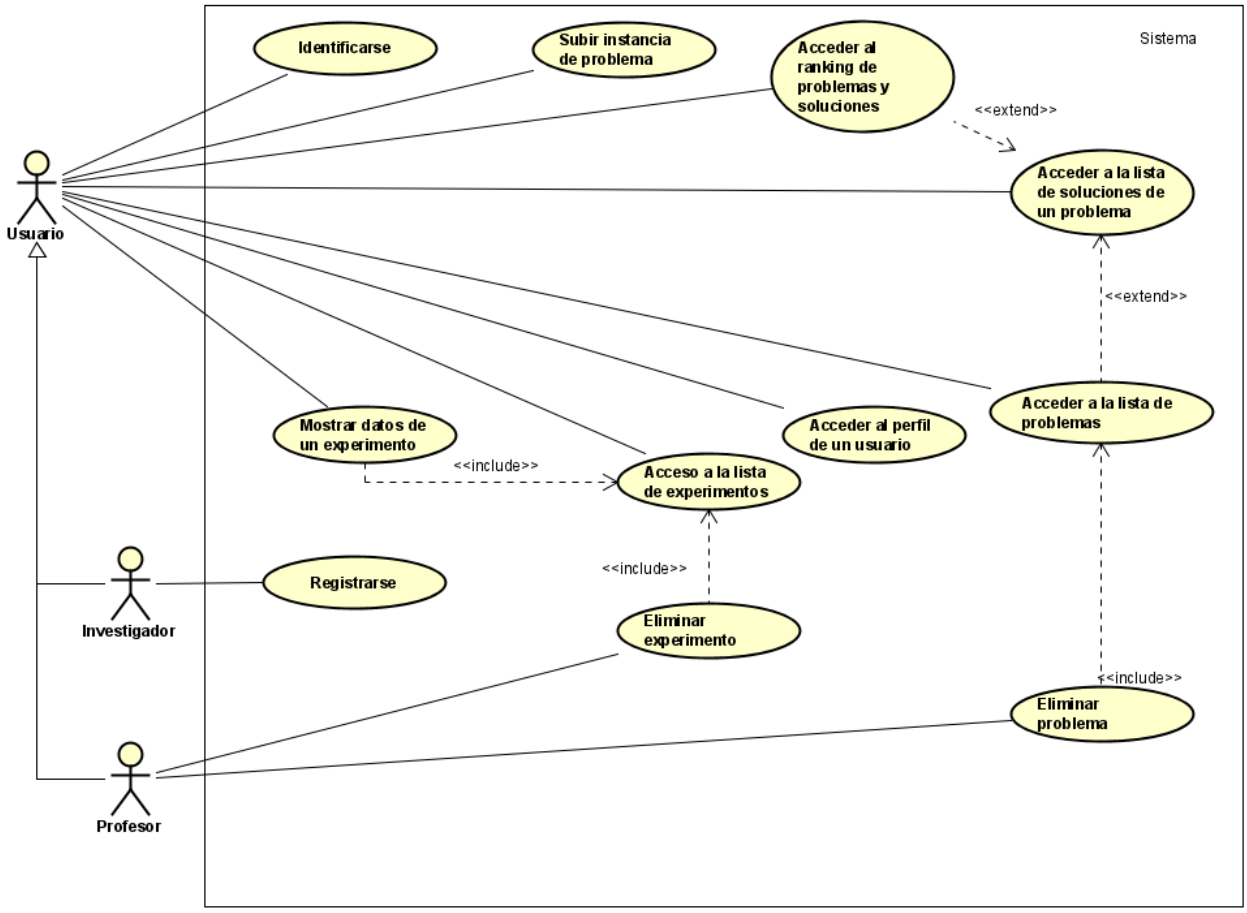

Figura 4.2: Diagrama de casos de uso

Capítulo 5

Diseño

5.1 Arquitectura del sistema

5.1.1 Patrón MVC

El patrón MVC se trata de una arquitectura del software que se usa para separar código según sus responsabilidades, manteniendo distintas capas que se encargan de hacer una tarea concreta.

Se usa sobre todo en sistemas donde se requiere el uso de interfaces de usuario. Su fundamento es la separación del código en tres capas diferentes en función de su responsabilidad. Estas capas se llaman modelo, vista y controlador.

En la capa de modelo se encuentra la representación de los datos de la aplicación, es decir, las entidades que nos servirán para poder guardar la información del sistema que se está desarrollando. Esta capa es la responsable de que el sistema se encuentre en un estado consistente e íntegro. Por último, la capa de modelo también es la encargada de gestionar el almacenamiento y recuperar los datos y entidades del dominio, es decir, que incluirá mecanismos de persistencia o será capaz de interactuar con ellos.

Los componentes de la capa de la vista son los encargados de generar las interfaces de nuestra aplicación. Se suele decir que la vista es una representación del estado del modelo en un momento concreto. Cuando las vistas componen la interfaz de usuario, deberán contener elementos de interacción que permitan al usuario enviar información e invocar acciones en el sistema.

Por último, la capa del controlador es la encargada de hacer de intermediario entre el usuario y el sistema. Será capaz de, a partir de una acción del usuario sobre la vista, interpretarla y actuar en consecuencia. Realiza también una transformación de datos para hacer que los componentes de la vista y el modelo se entiendan. Por tanto, se puede considerar que esta capa es un coordinador general del sistema [23].

Por tanto, como nuestro servicio necesita interfaces de usuario, un sistema de almacenamiento persistente de datos y un intermediario que interprete las acciones de los usuarios y el sistema de almacenamiento de datos, se va a aplicar el modelo desarrollado anteriormente siguiendo la imagen 5.1.

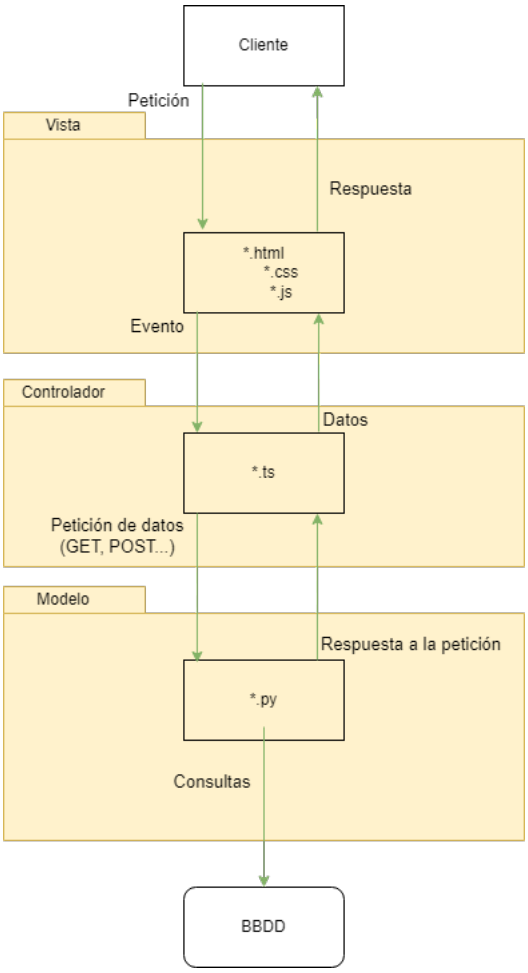

Figura 5.1: Diseño por capas siguiendo el patrón MVC

5.1.2 Diagrama de despliegue

Un diagrama de despliegue es un tipo de diagrama UML en el que se muestra la arquitectura de ejecución de un sistema. Se utilizan para visualizar el hardware y el software de un sistema, por lo que es necesario plantearlo al principio del diseño.

A partir de la figura 5.2 se puede ver que tenemos un nodo que será el dispositivo del usuario, su ordenador, a través del cual accederá a nuestro servicio mediante peticiones HTTP. A través de estas peticiones podrá acceder a la API que es quien realiza las operaciones sobre la base de datos. El usuario no será capaz de interactuar directamente sobre la API, sino que sólo lo hará sobre nuestro servicio de *Front-end*, desarrollado usando Angular, donde se encuentran las operaciones necesarias programadas para que pueda acceder a los datos y mostrárselos al usuario.

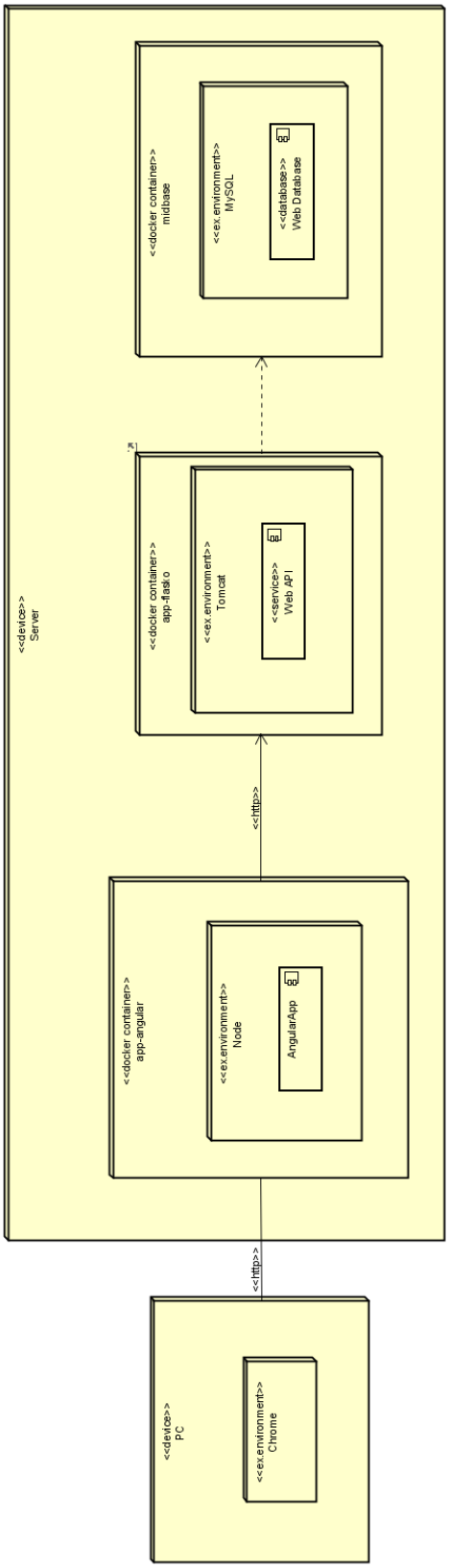

Figura 5.2: Diagrama de despliegue

5.1.3 Modelo de dominio

El modelo de dominio 5.3 muestra las diferentes clases u objetos con los que el proyecto trata, y la información de cada uno.

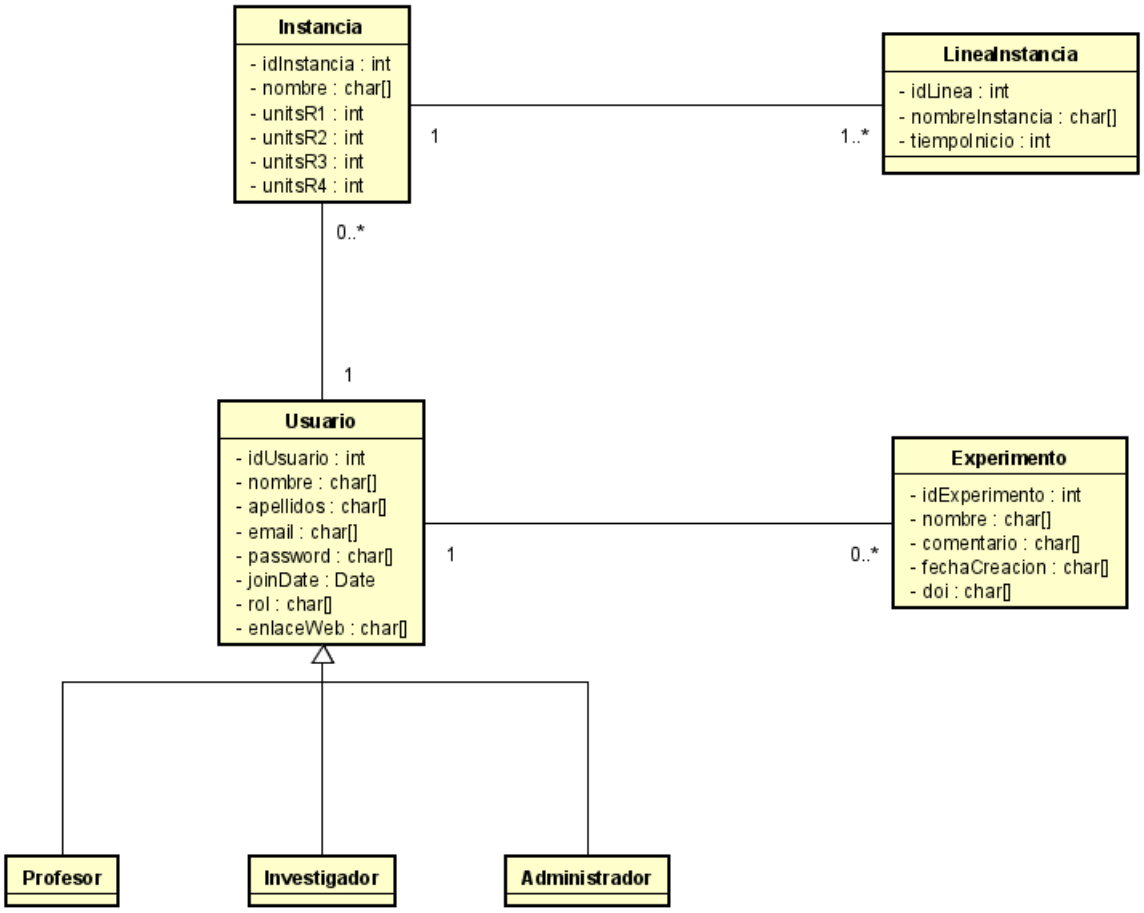

Figura 5.3: Diagrama de clases

5.1.4 Modelo de entidad relación

El modelo de entidad relación de la figura 5.4 muestra las tablas que tiene la base de datos del proyecto y sus campos, donde se guardarán toda la información necesaria.

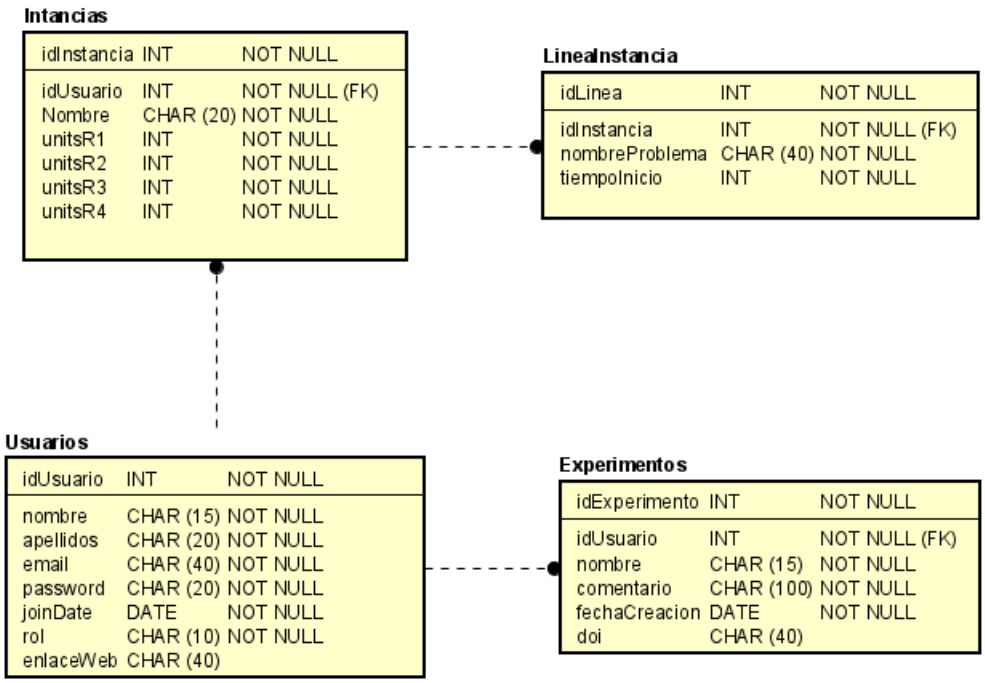

Figura 5.4: Diagrama de entidad relación de la base de datos

Capítulo 6

Implementación

6.1 Herramientas de Desarrollo

Esta sección presenta todas las herramientas que han sido utilizadas durante todas las etapas del desarrollo del proyecto, como el análisis, diseño, implementación y documentación.

6.1.1 Trello

Trello [24] es una herramienta sencilla y flexible que permite gestionar las tareas de un proyecto de manera visual. Las tareas se organizan en tableros para poder ver de manera clara las tareas pendientes, en progreso o terminadas entre otros estados.

Se ha usado esta página con el fin de controlar el progreso del proyecto y las actividades que componen todos los incrementos.

6.1.2 Overleaf

Overleaf [25] es una página web que permite crear documentos de alta calidad. Los documentos se guardan en la nube para así poder editar y guardarlos cuando se quiera. También existen guías y ayudas que permiten, mediante comandos, completar y personalizar el documento a placer del escritor.

Se ha utilizado esta página web para realizar la memoria de este Trabajo de Fin de Grado, editando el documento con LaTeX [26], permitiendo personalizar y adecuar el formato de éste.

6.1.3 Astah Professional

Astah Professional [27] es una herramienta que permite crear diagramas UML [28] para el diseño y análisis de proyectos. Además de modelado, tiene la funcionalidad de ingeniería inversa, generación de código y exportar documentos.

En esta herramienta se ha desarrollado la mayor parte del análisis y diseño del proyecto, en concreto el diagrama de casos de uso de la figura 4.2, las listas de requisitos de las tablas 4.1, 4.2 y 4.3, además del diagrama de despliegue de la figura 5.2, el diagrama de clases de la figura 5.3 y el diagrama entidad relación de la figura 5.4.

6.1.4 Visual Studio Code

Visual Studio Code [29] es un editor de código fuente. Permite la escritura, depuración y refactorización de código junto con la creación de atajos de teclado para personalizar su uso.

Se ha utilizado esta herramienta para generar y editar todos los ficheros de código necesarios para que el proyecto funcionase. También se han utilizado extensiones para facilitar el control de los archivos y herramientas Docker.

6.1.5 Draw.io

Draw.io [30] es una página web es una herramienta que permite crear diagramas UML, diagramas de entidad relación, diagramas de flujo, etc, ya que tiene una interfaz que nos permite editar las imágenes a nuestra elección.

Permite crear los diagramas y almacenarlos en la nube o en nuestro dispositivo para poder editarlas más tarde. También permite exportar los diagramas en diferentes formatos para poder incluirlos en documentos. Se ha utilizado esta página web para realizar el diagrama de capas que sigue este proyecto de la figura 5.1.

6.1.6 Angular

Angular [31] es un *framework* de código abierto orientado al diseño de aplicaciones web de una sola página que permite crear aplicaciones de una o varias páginas. Es una herramienta que permite generar código gracias a las plantillas que nos ofrece y después poder modificar para cumplir con nuestras necesidades.

6.1.7 Typescript

Typescript [32] es un lenguaje de programación de código abierto que proporciona varias mejoras con respecto a JavaScript [33] para proyectos de gran tamaño. Permite usar objetos basados en interfaces que permite tener más control sobre los datos que se tratan en los proyectos que se ejecutarán en el lado del cliente o del servidor.

6.1.8 HTML y CSS

HTML [34] es el lenguaje en el que se elaboran las páginas web. Es un lenguaje que permite identificar la estructura del documento mediante etiquetas. Al ser un estándar, HTML es un lenguaje que permite que cualquier página web escrita en una versión determinada pueda ser interpretada de la misma forma por cualquier navegador web actualizado.

CSS [35] es un lenguaje de diseño gráfico que permite definir y crear la presentación de un documento estructurado, como es HTML. Está diseñado principalmente para modificar la forma de presentación como las capas, *layouts*, colores o fuentes.

6.1.9 Bootstrap

Bootstrap [36] es una biblioteca para diseño de aplicaciones web que contiene muchas plantillas de diseño como formularios, botones, menús de navegación, etc. Todo basado en HTML y CSS que permite añadir nuevos estilos a las interfaces convencionales.

6.1.10 Docker

Docker [37] es un proyecto de código abierto que sirve para automatizar el despliegue de aplicaciones dentro de contenedores de software. Esto proporciona una capa adicional de abstracción y automatización de virtualización de aplicaciones.

6.1.11 Nginx

Nginx [38] es un servidor web/proxi inverso ligero de alto rendimiento. Es un software libre y de código abierto que se encarga de la entrega de aplicaciones web respondiendo a peticiones HTTP realizadas por los usuarios.

6.1.12 Ubuntu

Ubuntu [39] es una distribución Linux basada en Debian GNU/Linux, que incluye principalmente software libre y de código abierto. Puede utilizarse en ordenadores y servidores.

Es una bifurcación del código base del proyecto Debian GNU/Linux. El objetivo inicial era hacer de Debian una distribución más fácil de usar y entender para los usuarios finales, corrigiendo varios errores de este y haciendo más sencillas algunas tareas como la gestión de programas.

6.1.13 MySQL y SQLAlchemy

MySQL [40] es un sistema de gestión de bases de datos que cuenta con una doble licencia. Por una parte es de código abierto, pero por otra, cuenta con una versión comercial gestionada por la compañía Oracle. MySQL presenta algunas ventajas que lo hacen muy interesante para los desarrolladores.

La más evidente es que trabaja con bases de datos relacionales, es decir, utiliza tablas múltiples que se interconectan entre sí para almacenar la información y organizarla correctamente.

SQLAlchemy [41] es un kit de herramientas SQL de código abierto para el lenguaje de programación Python. Facilita la lectura y escritura en bases de datos relacionales a medida que su escala aumenta.

6.1.14 Python

Python [42] es un lenguaje de programación de alto nivel que puede ser utilizado para desarrollar aplicaciones de todo tipo. Es un lenguaje de programación que soporta la orientación a objetos y la programación imperativa y funcional.

6.1.15 Flask

Flask [43] es un *framework* escrito en Python que ofrece herramientas para crear aplicaciones web usando Python. Es un *framework* fácil de usar para los programadores ya que permite crear una aplicación web utilizando únicamente un archivo Python.

6.1.16 Postman

Postman [44] es una herramienta gratuita para poder ejecutar peticiones HTTP. Suele usarse para pruebas de una API, ya que permite realizar peticiones HTTP personalizadas a cualquier dirección pudiendo modificar cabeceras, *cookies* e incluso el cuerpo de la petición.

6.2 Implementación

Se va a explicar en puntos generales lo que ha sido necesario desarrollar para poder conseguir que el servicio web funcione correctamente.

6.2.1 Back-end

Para conseguir una funcionalidad válida, se ha necesitado implementar una API-REST usando Python con Flask, para aligerar y facilitar el trabajo, junto con una base de datos usando MySQL y SQLAlchemy para poder

almacenar los datos introducidos por los usuarios y que otros puedan acceder a ellos mediante las correspondientes peticiones

Base de datos

Como se ha indicado en el diagrama entidad relación de la figura 5.4, la base de datos está compuesta por cuatro tablas, a saber, Usuarios, Instancias, LineaInstancia y Experimentos.

La tabla Usuarios, tal y como se ve en la figura 6.1 almacena la información correspondiente a los usuarios registrados en el sistema. Como se ha mostrado, tiene los siguientes campos:

- idUsuario: Es la clave primaria de la tabla, es un campo que se autoincrementa cada vez que se hace una inserción en la tabla, por lo que no tenemos que preocuparnos por controlar el valor de este campo al insertar datos.
- Nombre: Nombre del usuario.
- Apellidos: Apellidos del usuario.
- Email: Dirección de correo electrónico del usuario que no se permitirán repetidos.
- Password: Contraseña del usuario
- Rol: El rol que tendrá el usuario en el sistema. Por defecto, un usuario tendrá el rol de ”Investigador” mientras que los profesores que gestionen la página tendrán el rol de ”Profesor”. Adicionalmente, como se tiene que seguir gestionando la página, manteniéndola e incluso ampliarla, habrá un usuario con el rol de ”Administrador”.
- JoinDate: Fecha en la que el usuario se ha registrado en formato AAAA-MM-DD.
- EnlaceWeb: Enlace a la página web personal del investigador. Este campo es opcional, por lo que el usuario podrá modificarlo una vez registrado si desea añadirla.

```python
#Nombre para la tabla de usuarios
__tablename__ = "usuarios"

#Atributos de la tabla de usuarios
idUsuario = db.Column(db.Integer, primary_key = True, autoincrement=True)
nombre = db.Column(db.String(30), nullable=False)
apellidos = db.Column(db.String(50), nullable=False)
email = db.Column(db.String(35), nullable=False, unique=True)
password = db.Column(db.String(25), nullable=False)
rol = db.Column(db.String(25), nullable=False)
joinDate = db.Column(db.Date(), nullable=False)
enlaceWeb = db.Column(db.String(100))
problemas = db.relationship('Instancia', backref='usuario')
experimentos = db.relationship('Experimentos', backref='usuario', lazy=True)
```

Figura 6.1: Declaración de los campos de la taba Usuarios en la base de datos

Como la tabla Usuarios tiene relaciones con otras tablas, se necesita añadir la indicación de que las tablas Instancia y Experimentos hacen referencia a entradas de esta tabla, por lo que se añade la opción *relationship* con el campo *backref* para indicarlo [45].

La tabla Instancia observando la figura 6.2 almacena la información correspondiente a las instancias subidas al sistema. Para poder guardar la información de las instancias, se han creado los siguientes campos en la tabla:

- idInstancia: Es la clave primaria de la tabla y un campo que se autoincrementa cada vez que un usuario añade una nueva instancia al sistema.
- idUsuario: Como se ha visto en la tabla Usuario, este atributo es una clave foránea que hace referencia al identificador del usuario que ha creado esta instancia.
- Nombre: Nombre de la instancia.
- unitsR1: Cantidad de unidades que utiliza la instancia del primer recurso.
- unitsR2: Cantidad de unidades que utiliza la instancia del segundo recurso.
- unitsR3: Cantidad de unidades que utiliza la instancia del tercer recurso.
- unitsR4: Cantidad de unidades que utiliza la instancia del cuarto recurso.

Adicionalmente, como la instancia tiene que guardar la información de cada problema, se ha creado una relación entre esta tabla y la tabla LineaInstancia. Esta vez se ha añadido la opción *cascade* para que cuando un profesor elimine una instancia, también se eliminen las entradas correspondientes en la tabla LineaInstancia.

```
#Nombre para la tabla de problemas
__tablename__ = "instancias"

#Atributos de la tabla de problemas
idInstancia = db.Column(db.Integer, primary_key=True)
idUsuario = db.Column(db.Integer, db.ForeignKey('usuarios.idUsuario'), nullable=False)
nombre = db.Column(db.String(50), nullable=False)
unitsR1 = db.Column(db.Integer, nullable=False)
unitsR2 = db.Column(db.Integer, nullable=False)
unitsR3 = db.Column(db.Integer, nullable=False)
unitsR4 = db.Column(db.Integer, nullable=False)
lineas_instancia = db.relationship('LineaInstancia', cascade='delete', backref='instancia', lazy=True)
```

Figura 6.2: Declaración de los campos de la taba Instancia en la base de datos

En la tabla LineaInstancia de la figura 6.3 se va a guardar la información relativa a cada uno de los problemas que se han añadido a una Instancia, por tanto se han creado los siguientes campos:

- idLinea: Clave primaria de la tabla, también es un campo que se autoincrementa, por lo que no necesitamos controlar su valor al insertar una nueva entrada.
- idInstancia: Como indicamos en la tabla Instancia, se tiene que guardar una referencia a la instancia a la que pertenece esta entrada, por lo que este valor será una clave foránea de la tabla Instancia.
- nombreProyecto: El nombre del proyecto que se añade a la instancia.
- tiempoInicio: El tiempo en el que el proyecto empieza a ejecutarse durante la simulación.

```
#Nombre para la tabla de problemas
__tablename__ = "LineaInstancia"

#Atributos de la tabla de problemas
idLinea = db.Column(db.Integer, primary_key=True, autoincrement=True)
idInstancia = db.Column(db.Integer, db.ForeignKey('instancias.idInstancia'), nullable=False)
nombreProyecto = db.Column(db.String(50), nullable=False)
tiempoInicio = db.Column(db.Integer)
```

Figura 6.3: Declaración de los campos de la taba LineaInstancia en la base de datos

Por último, en la tabla Experimentos, como se puede ver en la figura 6.4 se guarda la información relativa a los experimentos que los usuarios suben al sistema, por lo que se han creado los siguientes campos:

- idExperimento: Identificador único del experimento. Clave primaria de la tabla que se autoincrementa cada vez que se añade un nuevo experimento al sistema.
- idUsuario: Como se vio en la tabla Usuario, es el identificador del usuario que ha creado el experimento, por lo que es una clave foránea.
- Nombre: Nombre del experimento.
- Comentario: Comentario explicativo que el usuario añade al experimento.
- FechaCreación: Fecha en la que el experimento se ha añadido al sistema.
- DOI: Enlace al DOI del experimento, si es que existe.

```
__tablename__ = "Experimentos"


idExperimento = db.Column(db.Integer, primary_key=True, autoincrement=True)
idUsuario = db.Column(db.Integer, db.ForeignKey('usuarios.idUsuario'), nullable=False)
nombre = db.Column(db.String(15), nullable=False)
comentario = db.Column(db.String(100), nullable=False)
fechaCreacion = db.Column(db.Date, nullable=False)
doi = db.Column(db.String(50))
```

Figura 6.4: Declaración de los campos de la taba Experimentos en la base de datos

Para poder conectar esta base de datos a la API que se utilizará, es necesario añadir unas lineas al principio de ese archivo como se ve en la figura 6.5. Será necesario indicar el nombre de usuario, contraseña, *host* y nombre del *host* para poder conectarse a esta base de datos. También hay que asegurarse de que las tablas estén creadas, por lo que en la primera llamada que se haga desde la API, se hará una llamada a la función *create_all()* que ofrece la librería SQLAlchemy.

API-REST

Una API (*Application Programming Interface*) es un conjunto de reglas definidas para especificar cómo va a ser la comunicación entre dos componentes software, en nuestro caso, nuestra base de datos y la interfaz de usuario.

```python
def create_app():
    app = Flask(__name__)
    CORS(app)
    app.config['JWT_SECRET_KEY'] = [redacted] # Clave secreta para firmar los tokens JWT
    jwt = JWTManager(app)

    app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://{}:{}@{}/{}?charset=utf8mb4'.format(
        os.getenv('DB_USER', 'flask'),
        os.getenv('DB_PASSWORD', [redacted]),
        os.getenv('DB_HOST', 'mysql'),
        os.getenv('DB_NAME', 'flask')
    )

    return app

app = create_app()

api = Api(app)
db.init_app(app)

# create the DB on demand
@app.before_first_request
def create_tables():
    db.create_all()
```

Figura 6.5: Instrucciones para enlazar la API con la BBDD del sistema

Una API-REST es una interfaz de comunicación entre sistemas que usa el protocolo de transferencia de hipertexto, HTTP, para obtener datos o ejecutar las operaciones definidas sobre los datos de las bases de datos.

Es un sistema basado en el modelo cliente-servidor, donde el cliente, la interfaz de usuario, solicita obtener los recursos o realizar alguna operación sobre los datos almacenados. Mientras que el servidor, la base de datos, es quien entrega o procesa los datos que ha solicitado el cliente.

Con el fin de proteger y controlar el acceso a los datos, se ha implementado un método de validación de sesión utilizando tokens JWT (JSON Web Token). Gracias a la librería *jwt* [46] de Python, se ha podido utilizar esta funcionalidad de una manera mucho más sencilla. Usando funciones definidas en ésta como *get_jtw_identity()* y conociendo la estructura del token, se puede conocer quién está intentando acceder a los datos.

Junto con esta función, se ha utilizado el decorador *@jwt_required()* en las funciones necesarias para que, con el fin de evitar que cualquier usuario acceda a los datos, la petición requiera de un token JWT válido en la cabecera para poder realizarse correctamente. Por lo tanto, con estas dos funcionalidades de la librería *jwt* podemos comprobar si el usuario se ha identificado en el servicio, ya que controlamos la generación de estos tokens en la identificación del usuario, y cuando sea necesario podremos comprobar de qué tipo de usuario se trata para poder permitir que vea más información.

Este método de validación se ha implementado de la siguiente manera:

- Las funciones que requieren control sobre el acceso a los datos, como algunos métodos POST, los métodos DELETE, algunos PUT y los métodos GET que requieren sacar información especial de los datos tienen el decorador *@jwt_required()*.

- Antes de solicitar cualquier información a la base de datos, se valida el token que ha recibido la API, por lo que, mediante la función *get_jwt_identity()* que devuelve la identidad del token, que en este caso es el *email* del usuario, validamos si el usuario se encuentra en la base de datos, es decir, es un usuario que está registrado.

Como se puede ver en la figura 6.6, ya que utilizamos el *email* como uno de los identificadores del usuario, estamos seguros de que, de devolver algún usuario que cumpla esa *query*, únicamente devolverá un usuario que será quien esté intentando realizar esa operación.

```python
@app.route('/instancias', methods=['POST'])
@jwt_required()
def crear_instancia():
    # Obtener el email del usuario a partir del token JWT
    email = get_jwt_identity()

    # Buscar el usuario en la base de datos por su email
    usuario = Usuario.query.filter_by(email=email).first()

    if not usuario:
        return {'message': 'Usuario no encontrado'}, 404
```

Figura 6.6: Validación del usuario que realiza la petición mediante token JWT

Con el fin de gestionar los usuarios de la base de datos, se han implementado las operaciones básicas de una API-REST, que son GET, POST, PUT y DELETE. En cada uno de ellos, si fuese necesario, validando tanto el token JWT que pueda contener la petición como los datos de ésta.

En el caso de la petición GET para obtener todos los usuarios, como lo único que necesitamos es que nos devuelva la lista completa de usuarios no necesitaremos ninguna condición, por lo que realizamos una *query* para obtener una lista con todos los usuarios de la base de datos y se devuelve.

En el caso de que la petición GET sea para obtener los datos de un único usuario, la *query* que se realiza es utilizando el atributo idUsuario que se recibe por la URL, para así obtener los datos de ese usuario si existiera.

En el caso de una petición POST no es necesario un token JWT, ya que si es un nuevo usuario el que quiere registrarse en el servicio no va a poder tener acceso aún a uno. Lo que si se va a requerir es un JSON con los datos del usuario que se quiere registrar. Sin embargo, antes de poder introducir sus datos en la base de datos, es necesario que pase unas comprobaciones para poder comprobar la validez de los datos como se puede ver en la figura 6.7.

De la misma forma, las operaciones PUT y DELETE están protegidas de manera que la petición necesita de un token JWT para poder realizarse. También se validan los datos necesarios para realizar correctamente la petición, de lo contrario, la API devolverá el código de error 401, indicando que no se está autorizado a realizar esa operación en los datos.

Por último, para permitir a un usuario identificarse en el sistema, se tiene una petición POST que requiere del *email* y contraseña del usuario para identificarse. A partir de esos datos, si se encuentra un usuario en la base de datos con esos datos en los campos correspondientes, se procede a crear un token JWT como se ve en la figura 6.8 para permitir al usuario acceder a los datos pertinentes y poder realizar sus investigaciones.

```python
@app.route('/usuarios', methods=['POST'])
def nuevo_usuario():
    json = request.get_json(force=True)

    if json.get('nombre') is None or json.get('apellidos') is None or json.get('email') is None or json.get('password') is None:
        return {'message':'Bad request'}, 401

    if not re.match(r'^[\w\.-]+@[\w\.-]+\.\w+$',json.get('email')):
        return {'message':'Bad request'}, 401

    if json.get('nombre') is '' or json.get('apellidos') is '' or json.get('email') is '' or json.get('password') is '':
        return {'message':'Bad request'}, 401


    # Verificar si el correo electrónico ya existe en la base de datos
    existe_usuario = Usuario.query.filter_by(email=json.get('email')).first()
    if existe_usuario:
        return {'message': 'El correo electrónico ya está en uso'}, 401
```

Figura 6.7: Validación de los datos para la creación de un usuario

```python
payload = {"sub": email, "username": usuario.nombre}
secret_key = [redacted]
algorithm = "HS256"
expires_at = datetime.datetime.now() + datetime.timedelta(hours=2)

token = pyjwt.encode(
    {"exp": expires_at, **payload},
    secret_key,
    algorithm=algorithm
)
```

Figura 6.8: Creación del token JWT con los campos necesarios y la clave para cifrarlo

Para tratar las instancias, como ha sido necesario la implementación de una tabla auxiliar para indicar cada uno de los proyectos que contienen las instancias, se necesita la creación de los objetos con los que tratará el controlador para el correcto funcionamiento del servicio. Por esto, la petición GET en este caso es más compleja que la de los usuarios, ya que requiere realizar una *query* juntando dos tablas, la tabla Instancias y la tabla LineaInstancia mediante un *natural join*.

Esta operación nos permite obtener no solo todas las instancias, sino también todas las entradas de la tabla LineaInstancia que están relacionadas con cada una de ellas. Siguiendo la figura 6.9, primero se crea la lista de proyectos que contiene cada instancia y después se añaden todos los datos a un JSON para que el controlador lo procese y utilice como necesite.

Para añadir una instancia a la base de datos, lo primero es validar el usuario que está intentando realizar la petición. Si se trata de un usuario válido, se procede a crear las entradas en las tablas correspondientes para guardar la información. Como se puede ver en la figura 6.10, se obtiene la lista de unidades de recursos que se van a usar y se amplia hasta tener datos de los 4 recursos disponibles. A continuación, se crea la entrada en la tabla Instancias y, por cada entrada en la lista de proyectos, se crea una entrada en la tabla LineaInstancia que relacione uno de los proyectos permitidos con la instancia recién creada y el tiempo de inicio que tendrá ese proyecto en la simulación.

Para ver los datos de una instancia en concreto, se ha implementado una petición GET que requiere en la URL el atributo idInstacia. Con ese dato se procede a obtener todas las instancias y las entradas correspondientes de la tabla LineaInstancia, para generar un JSON. Éste mostrará todos los datos de la instancia para que los investigadores puedan ver su contenido y poder estudiarla.

```python
@app.route('/instancias', methods=['GET'])
def get_instancias():


    instancias = Instancia.query.join(LineaInstancia).all()
    instancias_json = []

    for instancia in instancias:
        proyecto_lista = []
        for linea_instancia in instancia.lineas_instancia:
            proyecto = {
                'filename': linea_instancia.nombreInstancia,
                'start': linea_instancia.tiempoInicio
            }
            proyecto_lista.append(proyecto)

        instancia_json = {
            'id': instancia.idInstancia,
            'idUsuario': instancia.idUsuario,
            'nombre': instancia.nombre,
            'projectList': proyecto_lista,
            'resources' : [instancia.unitsR1, instancia.unitsR2, instancia.unitsR3, instancia.unitsR4]
        }
        instancias_json.append(instancia_json)

    return {'instancias': instancias_json}, 200
```

Figura 6.9: Contenido de la petición para obtener todas las instancias

También se ha implementado la petición DELETE. Esta petición, como es importante ya que eliminaría una instancia de la base de datos, está protegida para que solamente los usuarios que sean profesores puedan usarla. Esto se hace comprobado el usuario que está identificado con el token JWT y buscando en la base de datos el rol de ese usuario que, si es diferente a el de un investigador, podrá ejecutarla.

```python
# Obtener datos de la lista enviada desde Angular
data = request.get_json(force=True)

resources = data['resources']
resources += [0] * (4-len(resources))

# Crear el problema y asociarlo al usuario
instancia = Instancia(data['nombre'], usuario.idUsuario, *resources)
db.session.add(instancia)
db.session.commit()

# Obtener el id del problema recién creado
id_instancia = instancia.idInstancia

# Insertar los registros en la tabla LineaProblema
for proyecto in data['projectList']:
    nombre_instancia = proyecto['filename']
    tiempo_inicio = proyecto['start']
    linea_instancia = LineaInstancia(idInstancia=id_instancia, nombreInstancia=nombre_instancia, tiempoInicio=tiempo_inicio)
    db.session.add(linea_instancia)

db.session.commit()
```

Figura 6.10: Creación de las entradas de datos necesarias para añadir una instancia

En cuanto a la gestión de experimentos, como queremos que todos los usuarios puedan ver la lista de los experimentos que los investigadores han publicado, se ha creado la petición GET de tal manera que no necesite un token JWT. Se realiza una *query* a la base de datos para obtener todos los experimentos y se realizará un *natural join* para también obtener la información del usuario que ha subido ese experimento. Con esos datos, se genera la lista de JSON que se devolverá al controlador para poder mostrar a todos los usuarios los datos de los experimentos.

Para publicar un experimento sí que es necesario estar identificado en el sistema, por lo que la petición POST pide un token JWT en la cabecera, con el que obtendrá el *email* del usuario para poder asignarle a éste el experimento que va a publicar. Como es posible que el experimento aún no esté publicado en una revista científica, no será necesario que el usuario introduzca un enlace al DOI del experimento, pero sí serán necesarios el resto de campos como se puede ver en la figura 6.11.

```python
@app.route('/experimentos', methods=['POST'])
@jwt_required()
def crear_experimento():
    datos = request.json

    if not comprobarJWT(get_jwt_identity()):
        return {'message': 'Usuario no encontrado'}, 401

    nombre = datos.get('nombre')
    comentario = datos.get('comentario')
    doi = datos.get('doi')

    if (nombre == '' or comentario == ''):
        return {'message':'Bad request'}, 401


    email = get_jwt_identity()
    usuario = Usuario.query.filter_by(email=email).first()
    idUsuario = usuario.idUsuario

    experimento = Experimentos(idUsuario, nombre, comentario)

    if (doi != ''):
        experimento.doi = doi

    db.session.add(experimento)
    db.session.commit()

    return {'message': 'Experimento creado correctamente'}, 201
```

Figura 6.11: Contenido de la petición para subir un experimento

Por último, para poder eliminar un experimento, se ha implementado la petición DELETE que requiere que el usuario esté identificado en el sistema por lo que pide que la cabecera tenga un token JWT. Además, para poder realizar la petición correctamente es necesario que el usuario que la realice sea un profesor, por lo que si el usuario no tiene el rol correspondiente, no se puede eliminar el experimento de la base de datos.

6.2.2 Front-end

Para que los usuarios puedan acceder a los datos del sistema e interactuar con ellos, se ha implementado una aplicación web basada en Angular ya que se ha trabajado con él anteriormente y es un *framework* adecuado para una aplicación web como esta.

Angular

Gracias a los comandos incorporados en Angular CLI [47] se han podido generar los componentes necesarios fácilmente y gracias a haber utilizado este *framework* anteriormente se ha evitado emplear tiempo en aprender a usar esta herramienta.

Como se ve en la figura 6.12, se han creado varios componentes para gestionar las vistas a las que los usuarios van a poder acceder. Gracias al comando *ng g c (nombre-Componente)* se han autogenerado los componentes con un archivo *.html, .css, .ts y .spec.ts* cada uno, que se han modificado para poder adaptarlos a la funcionalidad necesaria.

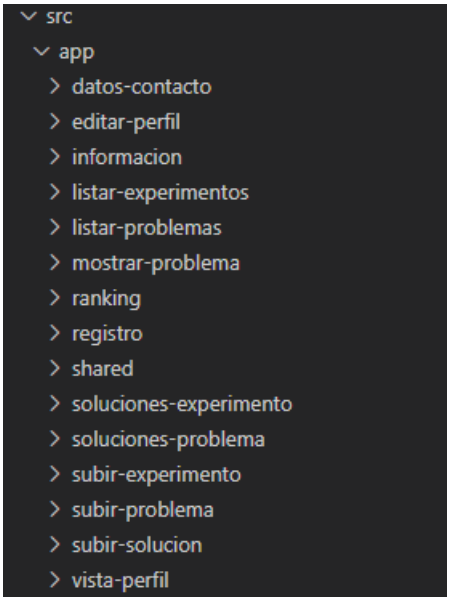

Figura 6.12: Listado de componentes de la aplicación Angular

Para poder acceder a todos estos componentes, Angular tiene un módulo que permite navegar entre todos estos componentes para encontrar la vista que se quiere mostrar. Este módulo se llama por defecto *AppModule* y funciona como el punto de entrada a la aplicación web, el módulo raíz. Este módulo se genera junto con la aplicación y tiene el aspecto de la figura 6.13.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Figura 6.13: Contenido del archivo app.module.ts por defecto

Se puede ver que tiene una lista llamada *declarations* en donde se van a indicar los componentes de la aplicación. Cada vez que se añade un nuevo módulo, se debe añadir a la lista, sino Angular devuelve un error.

La lista de *imports* contiene los módulos que los componentes necesitan para funcionar. Por ejemplo, si queremos que la aplicación realice peticiones HTTP, necesitará tener el módulo *HttpClientModule* que es quien configura las dependencias.

La lista *providers* indica los servicios que estarán disponibles para todos los componentes. Si un servicio no va ser necesario que se acceda desde todos los componentes, es preferible indicarlo sólo en aquellos que lo necesiten.

Por último, *bootstrap* contiene el componente raíz, el que se ejecutará al iniciar la aplicación web. En nuestro caso, como estamos definiendo los componentes en el módulo *AppModule*, será éste quien aparezca aquí.

Por tanto, teniendo en cuenta la funcionalidad de cada una de las listas, nuestro archivo deberá quedar como la figura 6.14, sin incluir los *imports* de cada módulo. Como necesitamos realizar formularios y peticiones HTTP se tendrán que añadir los correspondientes módulos a la lista de *imports*. También, como necesitamos controlar si el usuario está identificado al acceder a las vistas, se añade el servicio encargado de realizar las peticiones de los usuarios a la API en la lista de *providers* para que todos los componentes puedan acceder a ella.

```
@NgModule({
  declarations: [
    AppComponent,
    CHome,
    CProblemas,
    CSubirSolucion,
    CRanking,
    CSolucionesProblema,
    CMostrarProblema,
    CExperimentos,
    CSolucionesExperimento,
    CRegistro,
    CContacto,
    CSubirProblema,
    CVistaPerfil,
    CEditarPerfil,
    CSubirExperimento
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule
  ],
  providers: [
    ClientApiRestService,
    DataService
    ],

  bootstrap: [AppComponent]
})
export class AppModule { }
```

Figura 6.14: Contenido del archivo app.module.ts tras el desarrollo del proyecto

El *AppComponent* es el componente que está por encima de los demás, es decir, su archivo HTML se ejecutará por encima del del componente correspondiente. Por lo tanto, es aquí donde se incluirá la funcionalidad que queremos que se vea en todos los componentes. Una funcionalidad que se ha colocado aquí es la barra de navegación. Los botones que permiten al usuario ir a las diferentes opciones que ofrece la aplicación web se requiere que estén visibles en todas las vistas. También el control de la identificación o registro de un usuario, como es parte importante de la aplicación, pues con eso se controla qué funcionalidad puede utilizar el usuario, se colocará en este componente.

La barra de navegación se ha implementado usando la clase *navbar* presente en la librería Bootstrap. Se ha personalizado utilizando algunas variantes que ofrece esta clase junto con algunas modificaciones especificadas en el archivo CSS para poder conseguir una barra de navegación más accesible por el usuario.

Esta barra de navegación cuenta con dos vistas posibles, una cuando el usuario no está identificado como se ve en la figura 6.15 y otra cuando sí lo está como en la figura 6.16. Esto es para poder evitar que un usuario no identificado acceda a funcionalidad que no debe hasta que demuestre que es un investigador registrado en el sistema.

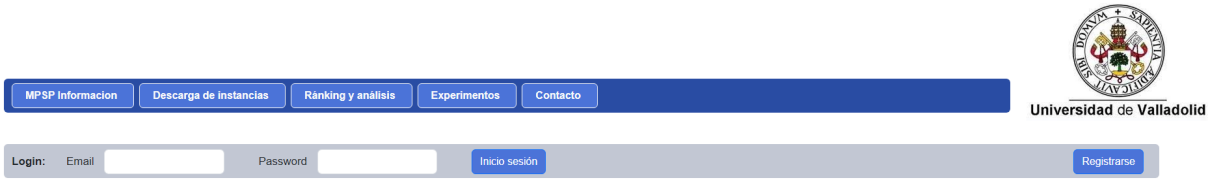

Figura 6.15: Vista de la barra de navegación de un usuario sin identificarse

Figura 6.16: Vista de la barra de navegación de un usuario identificado

Como se puede ver en ambas figuras 6.15 y 6.16, la funcionalidad de control de sesión también se ha incluido en otro *navbar* cuya vista cambia también si el usuario está identificado o no. En este caso, en vez de botones se ha usado un formulario para solicitar al usuario el *email* y su contraseña para iniciar sesión. También se ha añadido un botón para que, si el usuario aún no está registrado en el sistema, pueda acceder a la vista correspondiente.

Si el usuario está identificado, le mostrará un mensaje de bienvenida con el nombre del usuario y un botón donde antes estaba el de registrarse para cerrar la sesión. Esto recargará la página y se volverá a la vista de un usuario sin identificarse.

El componente de la página de inicio es el que muestra información sobre los RCMPSP y es la que se carga al entrar en la aplicación por defecto. Contiene una etiqueta *div* en el fichero HTML que se ha personalizado para que tenga la forma de la figura 6.17. Para seguir con la estética, se ha utilizado la misma etiqueta y colores en todas las vistas.

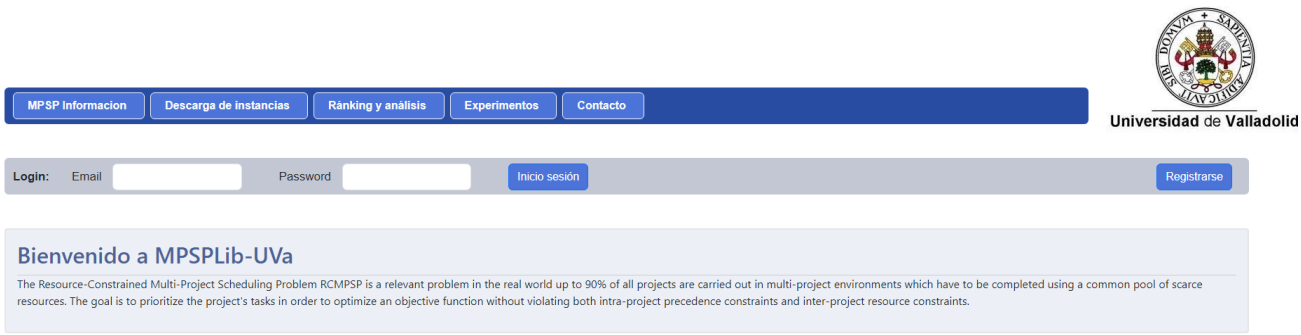

Figura 6.17: Interfaz completa de la vista cuando el usuario accede a la aplicación web

En el componente que permite ver la lista de instancias en el sistema se ha utilizado el mismo tipo de etiqueta *div* personalizada que en la vista del inicio, pero añadiendo una tabla en la que se muestran los datos al usuario.

Esta tabla está creada a partir de una lista que el archivo Typescript del componente obtiene de la base de datos. Lo que hace es pedir a la API la lista completa de los proyectos y, a partir de los campos que se han obtenido, se colocan los datos correspondientes en la tabla para su visualización.

Al cargar la vista, el archivo Typescript hará dos operaciones. La primera es pedir a la API la lista de proyectos y la otra será preguntar a la API, si el usuario está identificado, el rol de éste para decidir si mostrar lo que verá un investigador como se ve en la figura 6.18 o lo que verá un profesor como se ve en la figura 6.19.

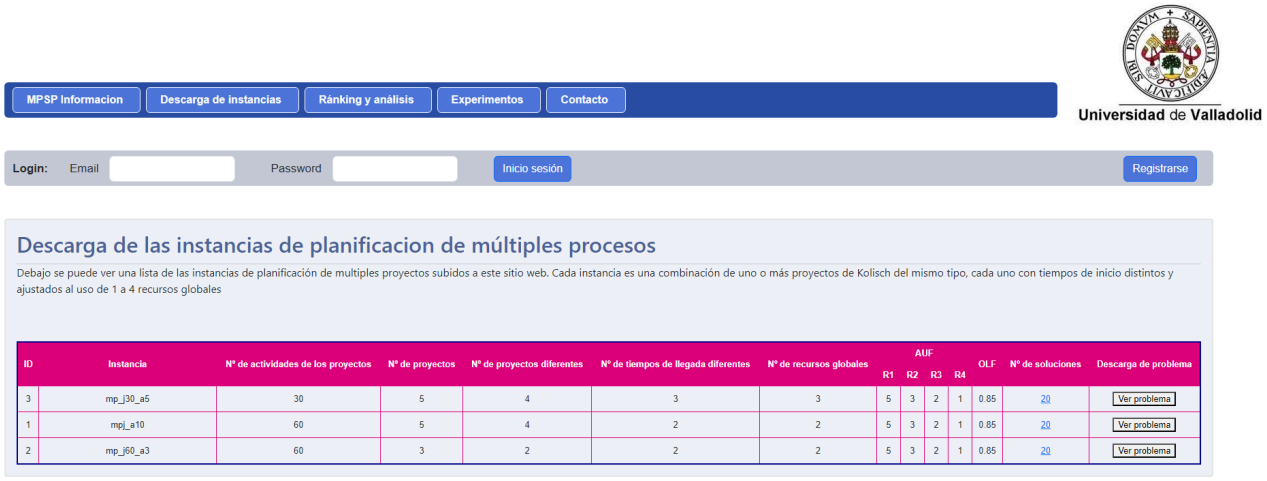

Figura 6.18: Interfaz de listado de instancias por usuario Investigador

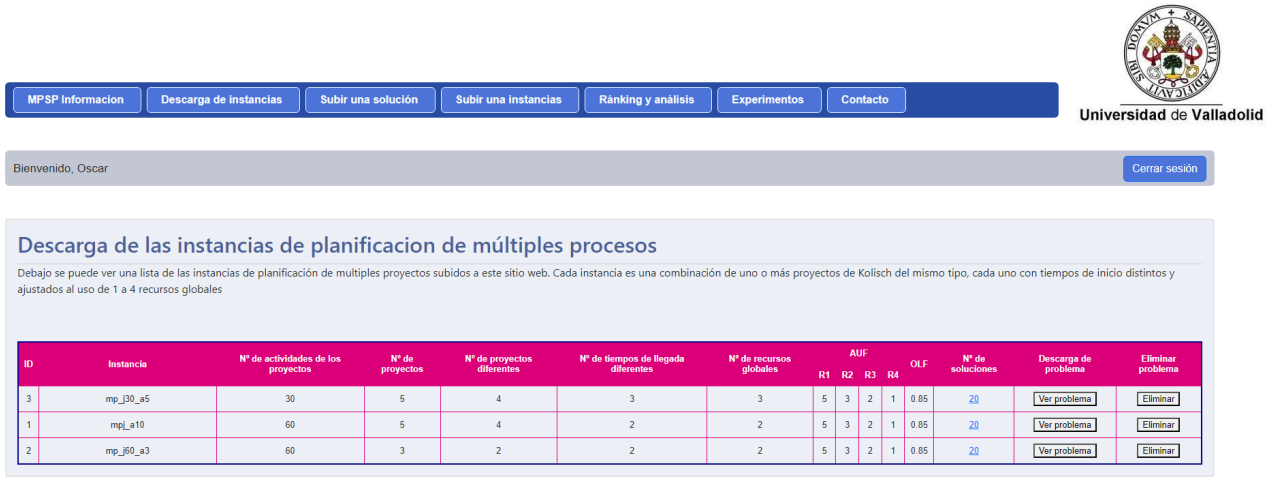

Figura 6.19: Interfaz de listado de instancias por un profesor

Como se puede ver, la información de las instancias se muestra de la misma manera para los dos tipos de usuarios. Sin embargo, al profesor se le da la posibilidad de, si ocurre cualquier problema con una instancia que un usuario ha subido, poder eliminarla del sistema.

Además, se muestra en el campo del número de soluciones junto con el valor de cada instancia, un enlace a la vista que mostrará la lista de soluciones de la instancia junto con su información.

En el componente para subir una solución, como aún no se ha desarrollado el sistema que valide y procese una solución, está inhabilitado para todos los usuarios como se puede ver en la figura 6.20. Se podrá acceder a esta vista y, pulsando en el texto indicado, se podrá ver un ejemplo de lo que será el formato de la solución pero no se permitirá subir soluciones al sistema todavía.

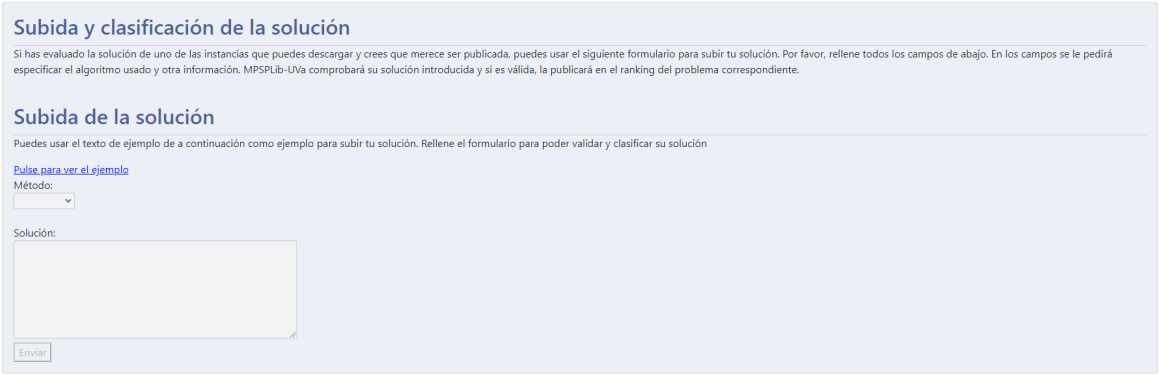

Figura 6.20: Interfaz de subida de solución inhabilitada

El componente encargado de la subida al sistema de nuevas instancias consta de un formulario interactivo que el usuario irá rellenando y, en consecuencia de lo que indique, se le pedirá la información necesaria para crear una nueva instancia.

Lo primero que se le pedirá será el nombre de la instancia como se ve en la figura 6.21. A continuación, se le pedirá que indique el número de trabajos que tendrá cada proyecto de entre los que hay guardados en el sistema. Una vez elegido el número de trabajos, se elegirá el número de proyectos que tendrá la instancia. En función del valor que haya indicado el usuario, se crearán tantas entradas para que el usuario elija los proyectos y tiempos de inicio de cada uno. Después se pedirán el número de recursos globales que usará la instancia y se tendrá que introducir un número del 1 al 4. De la misma manera, se crearán tantas entradas como el valor seleccionado para que el usuario introduzca las unidades de cada recurso que se usarán. Una vez hecho todo esto, si todo ha ido bien se mostrará un aviso de que se ha creado correctamente y se recargará la página como vemos en la figura 6.22.

Figura 6.21: Interfaz inicial para subir una instancia

Figura 6.22: Interfaz inicial para subir una instancia

Para ver el *ranking* de la mejor solución de cada instancia se ha creado otro componente. Éste utiliza el mismo formato de la tabla que en el componente del listado de instancias. En la parte izquierda de la tabla se mostrará algo de información de cada instancia y en esa misma fila a la derecha se mostrará la información de la mejor solución en función de las funciones objetivo que se desee optimizar que serán APD, TMS y DPD que son *Average Project Delay*, *Total Makespan* y *Standard Deviation of the Project Delay* respectivamente como se puede ver en la figura 6.23.

Cuando el usuario entra en la vista, el archivo Typescript solicita a la API la lista de instancias que hay en la base de datos. Cuando los obtiene, relaciona las soluciones con cada uno de las instancias en función del campo a optimizar, por lo que si se cambia el campo en el encabezado de la tabla, se podrían ver otras soluciones como la mejor para cada instancia.

Figura 6.23: Interfaz del ranking de soluciones de cada instancia

Si el usuario pulsa en el número de soluciones que tiene una instancia, se redireccionará a la vista de todas las soluciones de ésta. El componente se encarga de mostrar al principio la información de la instancia y debajo un listado de las soluciones con sus datos como se puede ver en la figura 6.24. Como aún no está implementada la funcionalidad de subir una solución en todas las figuras que se muestran de la aplicación web se han generado soluciones con datos aleatorios para poder simular esta funcionalidad.

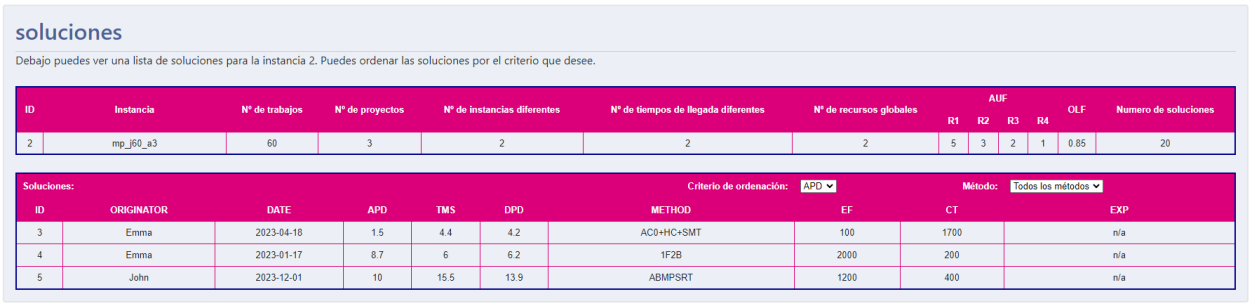

Figura 6.24: Interfaz de las soluciones de una instancia

Para ver el contenido de la instancia, es decir, los proyectos que la componen, sus tiempos de inicio y los recursos que utiliza, el usuario puede pulsar en el botón de la figura 6.9 que pone ”Ver problema”. En la vista se imprimirá usando etiquetas para que la información pueda entenderse más fácilmente como se puede ver en la figura 6.25.

```
<mp-list>
  <mp>
    <name>mp_j60_a3</name>
    <project-list>

        <project>
          <filename>j6010_7.sm</filename>
          <start>0</start>
        </project>

        <project>
          <filename>j6010_5.sm</filename>
          <start>0</start>
        </project>

        <project>
          <filename>j6010_7.sm</filename>
          <start>4</start>
        </project>

    </project-list>
    <resources>

        <resource>8</resource>

        <resource>9</resource>

        <resource>0</resource>

        <resource>0</resource>

    </resources>
  </mp>
</mp-list>
```

Figura 6.25: Interfaz de la vista del contenido de una instancia

Para poder visualizar todos los experimentos que los investigadores han subido al sistema, el componente que lo gestiona crea una tabla en la que se mostrarán los datos de cada una. Dependiendo del usuario que acceda a esta vista, puede variar el contenido de ésta. Por ejemplo, si un usuario que no se ha identificado accede a la lista de experimentos verá la información de cada uno como se puede ver en la figura 6.26. Si es un investigador que se ha identificado el que accede a esta vista, se le mostrará adicionalmente un botón debajo de la lista para poder subir al sistema un experimento como se ve en la figura 6.27. También si quien accede a esta vista es un profesor, tendrá la opción en la tabla de borrar el experimento del sistema como se puede ver en la figura 6.28.

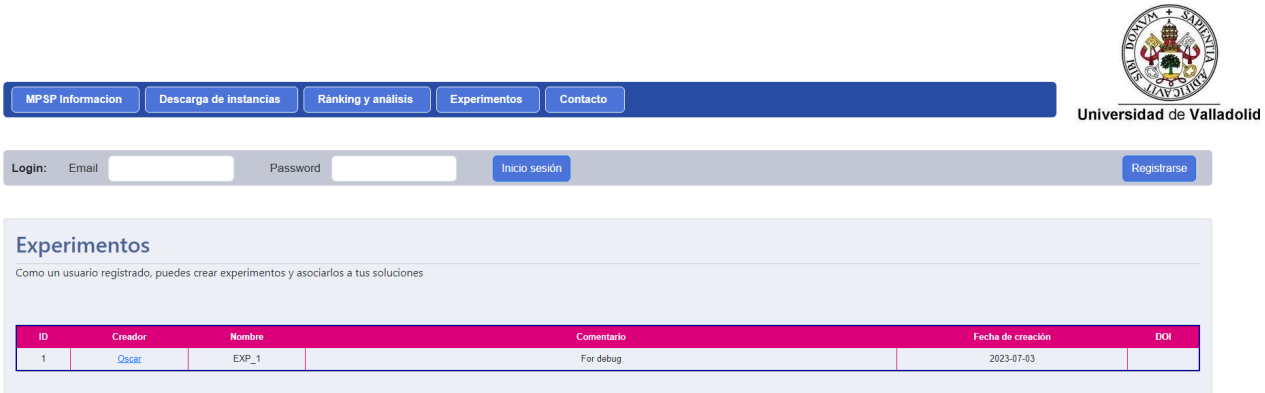

Figura 6.26: Interfaz de la vista de experimentos sin iniciar sesión

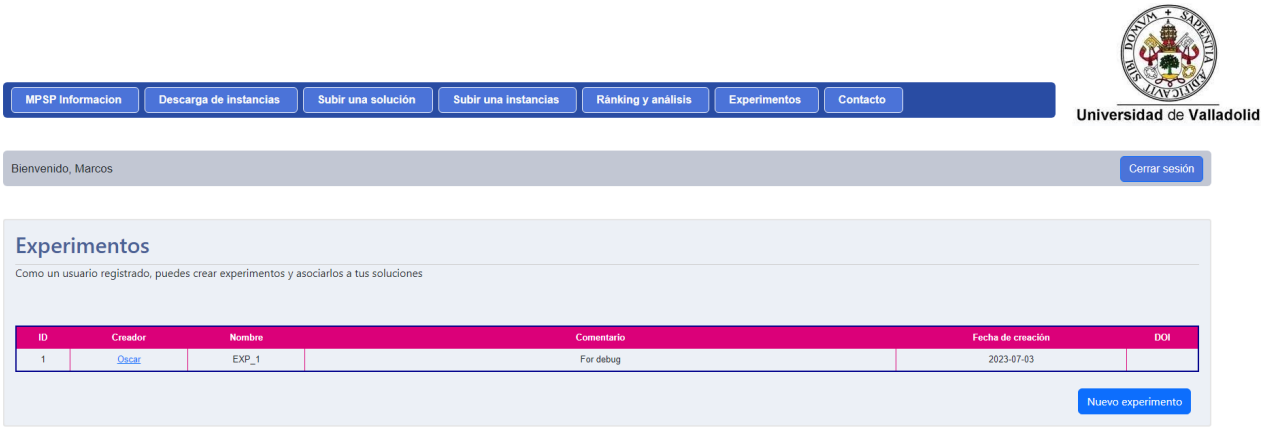

Figura 6.27: Interfaz de la vista de experimentos por un investigador

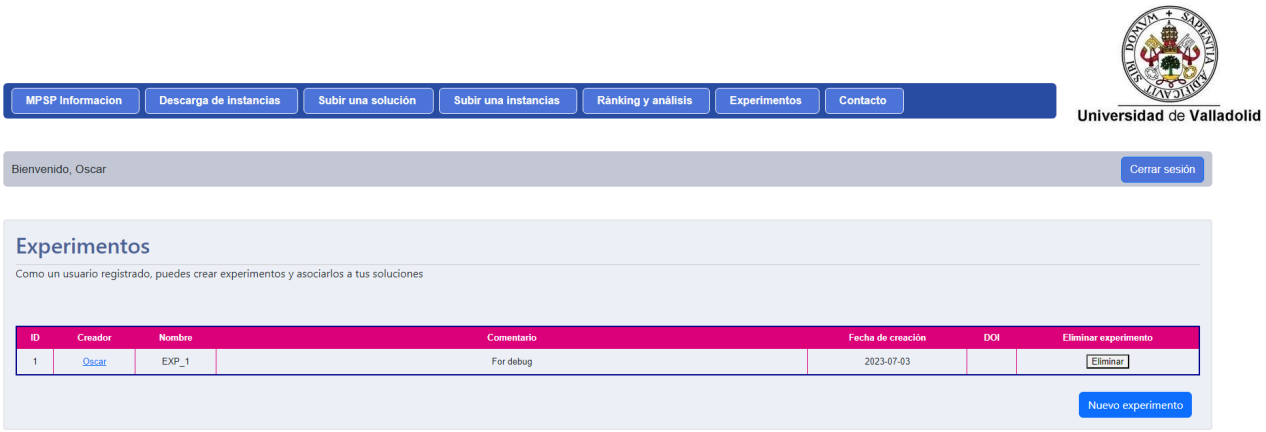

Figura 6.28: Interfaz de la vista de experimentos por un profesor

Cuando un investigador o profesor va a subir un nuevo experimento, se le piden el nombre, un comentario o descripción del experimento y el enlace al DOI, si lo tiene, como se ve en la figura 6.29. Los campos del nombre y comentario serán obligatorios para que se pueda aceptar el experimento, siendo el enlace al DOI un campo opcional. Una vez se pulse el botón para agregar el experimento, se mandará una petición a la API para añadirlo, relacionándolo con el usuario que lo haya subido.

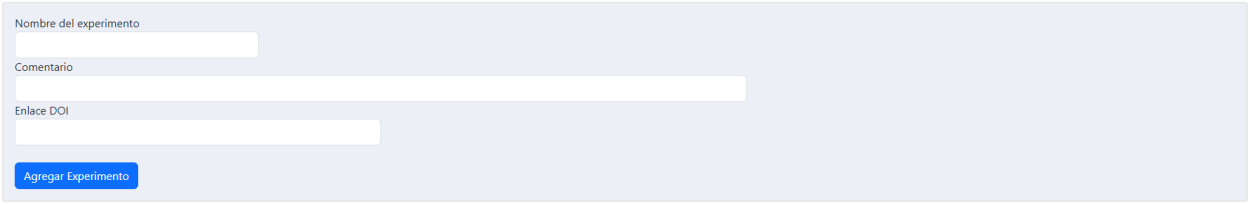

Figura 6.29: Interfaz de la vista para subir un nuevo experimento

Si el usuario pulsa en el botón de ”Contacto” de la barra de navegación, accederá a la vista que muestra la información de los profesores que dirigirán el servicio web. Como se puede ver en la figura 6.30 las investigadoras responsables de este servicio web van a ser la Dra. Elena Pérez y la Dra. Marta Posada. Se mostrarán los datos de dónde se encuentran actualmente dando docencia y sus *emails* a través de los que poder contactar si fuese necesario. Así mismo, se muestran los datos del desarrollador de la página web.

Figura 6.30: Interfaz de la vista con los datos de contacto de los profesores encargados de gestionar la aplicación web

Cuando se accede a la lista de experimentos, se puede ver el nombre del usuario que ha subido cada uno destacado. Esto indica que, si se pulsa sobre el nombre, el usuario será redireccionado a la vista con los datos de ese usuario. Se mostrarán el nombre, apellidos, *email*, fecha en la que se unió y si tiene página web como se ve en la figura 6.31.

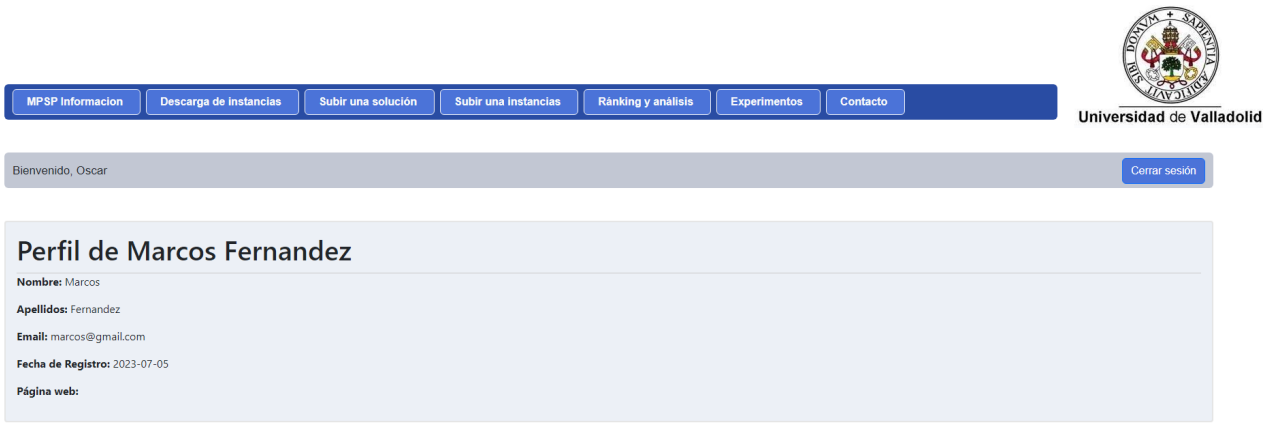

Figura 6.31: Interfaz de la vista del perfil de otro usuario

Adicionalmente, cuando se carga esta vista, el archivo Typescript pide junto a los datos del usuario que se compruebe, si está el usuario identificado, si se trata del mismo usuario que está identificado actualmente. En el caso de ser el mismo, se mostrará un botón para poder añadir o modificar la dirección de la página web del usuario tal y como se muestra en la figura 6.32.

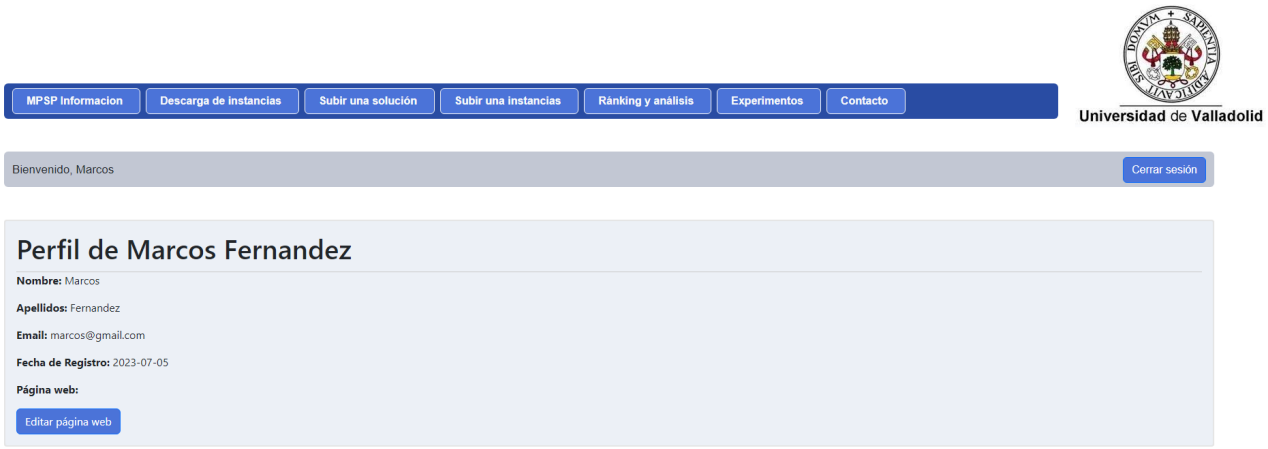

Figura 6.32: Interfaz de la vista del perfil del usuario identificado

Cuando un usuario vaya a modificar los datos de su perfil, la interfaz sólo le dejará modificar la página web del usuario según la figura 6.33. Con eso se permite que, si un investigador cambia de dirección web o antes no tenía y ahora sí, pueda compartirla con el resto de usuarios del sistema.

Por último, si el usuario introduce cualquier ruta que no se haya establecido en el componente *AppModule*, el servicio web lo redirigirá a la vista del inicio.

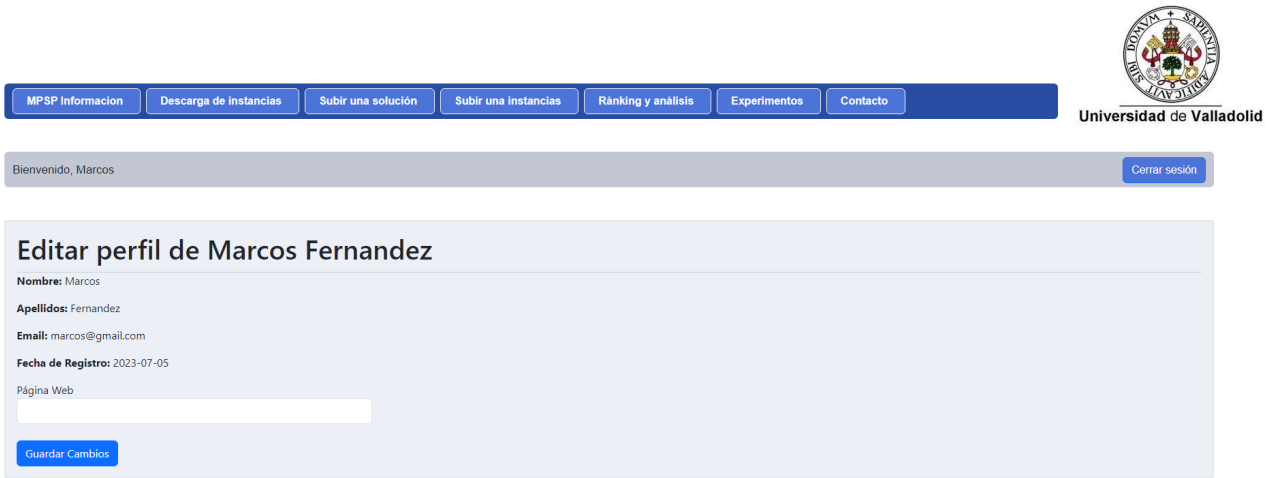

Figura 6.33: Interfaz de la vista para editar el perfil del usuario identificado

Capítulo 7

Pruebas

Durante el desarrollo del proyecto se han realizado pruebas que comprueban el buen funcionamiento de la aplicación. Estas pruebas incluyen llamadas a la API y pruebas en las interfaces para comprobar que toda funcionalidad está disponible para quien puede usarla.

Se ha seguido el modelo de caja negra para la realización de estas pruebas. Este modelo consiste en probar la aplicación mediante los datos introducidos y obtener un resultado que se comparará con el resultado esperado. Son un tipo de pruebas útiles para probar la funcionalidad del sistema y encontrar errores que se pueden haber pasado por alto a la hora de la implementación. Estas pruebas se realizan sin tener conocimiento del funcionamiento de la aplicación, por lo que sólo se tendrá en cuenta el resultado de la operación.

7.1 Pruebas de caja negra

A continuación, se van a detallar todas las pruebas realizadas, mostrándose una descripción del test, las acciones realizadas, precondiciones, el valor introducido en el sistema, el resultado esperado y el resultado obtenido, ver tablas 7.1 hasta la 7.30.

PCN - 01	**Registro de un usuario sin página web**
Descripción	El usuario introduce todos sus datos menos la página web y pulsa el botón de ”Registrarse” para añadir sus datos al sistema.
Acciones realizadas	1. El usuario pulsa el botón de ”Registrarse” bajo la barra de navegación 2. El usuario introduce sus datos en los campos dejando vacío el campo de la página web 3. El usuario pulsa el botón ”Registrarse” del formulario para enviar sus datos al sistema
Precondicion	Ninguna
Valor introducido	Nombre, apellidos, *email* y contraseña
Resultado esperado	Se añade el usuario a la base de datos, con el campo de la página web vacío y se muestra por pantalla un aviso de que el registro se ha creado correctamente
Resultado obtenido	Usuario creado con los campos introducidos y el campo de la página web vacío
Resultado de la prueba	Correcto

Tabla 7.1: Prueba de caja negra 1. Registro de usuario sin página web

PCN - 02	**Registro de un usuario con página web**
Descripción	El usuario introduce todos sus datos y pulsa el botón de ”Registrarse” para añadir sus datos al sistema.
Acciones realizadas	1. El usuario pulsa el botón de ”Registrarse” bajo la barra de navegación 2. El usuario introduce sus datos en los campos 3. El usuario pulsa el botón ”Registrarse” del formulario para enviar sus datos al sistema
Precondicion	Ninguna
Valor introducido	Nombre, apellidos, *email*, contraseña y página web
Resultado esperado	Se añade el usuario a la base de datos y se muestra por pantalla un aviso de que el registro se ha creado correctamente
Resultado obtenido	Usuario creado con los campos introducidos
Resultado de la prueba	Correcto

Tabla 7.2: Prueba de caja negra 2. Registro de usuario con página web

PCN - 03	**Registro de un usuario repitiendo correo**
Descripción	El usuario introduce todos sus datos pero introduce un correo que ya se encuentra en el sistema relacionado con otro usuario y pulsa el botón de ”Registrarse” para añadir sus datos al sistema.
Acciones realizadas	1. El usuario pulsa el botón de ”Registrarse” bajo la barra de navegación 2. El usuario introduce sus datos en los campos 3. El usuario pulsa el botón ”Registrarse” del formulario para enviar sus datos al sistema
Precondicion	Ya existe un usuario con ese correo en el sistema
Valor introducido	Nombre, apellidos, *email* y contraseña
Resultado esperado	Se detecta que ya existe un usuario con ese correo, se rechaza la solicitud de registro y se muestra un mensaje de error
Resultado obtenido	Petición rechazada y se muestra el mensaje de que ha habido un problema
Resultado de la prueba	Correcto

Tabla 7.3: Prueba de caja negra 3. Registro de usuario con un correo ya existente

PCN - 04	**Registro de un usuario sin nombre**
Descripción	El usuario introduce todos sus datos pero deja el campo del nombre vacío y pulsa el botón de ”Registrarse” para añadir sus datos al sistema.
Acciones realizadas	1. El usuario pulsa el botón de ”Registrarse” bajo la barra de navegación 2. El usuario introduce sus datos en los campos dejando vacío el campo del nombre 3. El usuario pulsa el botón ”Registrarse” del formulario para enviar sus datos al sistema
Precondicion	Ninguno
Valor introducido	Apellidos, *email* y contraseña
Resultado esperado	Se detecta que falta el valor del nombre, se rechaza la solicitud de registro y se muestra un mensaje de error
Resultado obtenido	Petición rechazada y se muestra el mensaje de que ha habido un problema
Resultado de la prueba	Correcto

Tabla 7.4: Prueba de caja negra 4. Registro de usuario sin nombre

PCN - 05	**Registro de un usuario sin apellidos**
Descripción	El usuario introduce todos sus datos pero deja el campo de los apellidos vacío y pulsa el botón de ”Registrarse” para añadir sus datos al sistema.
Acciones realizadas	1. El usuario pulsa el botón de ”Registrarse” bajo la barra de navegación 2. El usuario introduce sus datos en los campos dejando vacío el campo de los apellidos 3. El usuario pulsa el botón ”Registrarse” del formulario para enviar sus datos al sistema
Precondicion	Ninguno
Valor introducido	Nombre, *email* y contraseña
Resultado esperado	Se detecta que falta el valor de los apellidos, se rechaza la solicitud de registro y se muestra un mensaje de error
Resultado obtenido	Petición rechazada y se muestra el mensaje de que ha habido un problema
Resultado de la prueba	Correcto

Tabla 7.5: Prueba de caja negra 5. Registro de usuario sin apellidos

PCN - 06	**Registro de un usuario sin *email***
Descripción	El usuario introduce todos sus datos pero deja el campo del *email* vacío y pulsa el botón de ”Registrarse” para añadir sus datos al sistema.
Acciones realizadas	1. El usuario pulsa el botón de ”Registrarse” bajo la barra de navegación 2. El usuario introduce sus datos en los campos dejando vacío el campo del *email* 3. El usuario pulsa el botón ”Registrarse” del formulario para enviar sus datos al sistema
Precondicion	Ninguno
Valor introducido	Nombre, apellidos y contraseña
Resultado esperado	Se detecta que falta el valor del *email*, se rechaza la solicitud de registro y se muestra un mensaje de error
Resultado obtenido	Petición rechazada y se muestra el mensaje de que ha habido un problema
Resultado de la prueba	Correcto

Tabla 7.6: Prueba de caja negra 6. Registro de usuario sin *email*

PCN - 07	**Registro de un usuario sin contraseña**
Descripción	El usuario introduce todos sus datos pero deja el campo de la contraseña vacío y pulsa el botón de ”Registrarse” para añadir sus datos al sistema.
Acciones realizadas	1. El usuario pulsa el botón de ”Registrarse” bajo la barra de navegación 2. El usuario introduce sus datos en los campos dejando vacío el campo de la contraseña 3. El usuario pulsa el botón ”Registrarse” del formulario para enviar sus datos al sistema
Precondicion	Ninguno
Valor introducido	Nombre, apellidos y *email*
Resultado esperado	Se detecta que falta el valor de la contraseña, se rechaza la solicitud de registro y se muestra un mensaje de error
Resultado obtenido	Petición rechazada y se muestra el mensaje de que ha habido un problema
Resultado de la prueba	Correcto

Tabla 7.7: Prueba de caja negra 7. Registro de usuario sin contraseña

PCN - 08	**Inicio de sesión de un usuario**
Descripción	El usuario introduce sus datos bajo la barra de navegación para iniciar sesión en la aplicación web.
Acciones realizadas	1. El usuario introduce sus datos en los campos de la barra 2. El usuario pulsa el botón ”Inicio sesión” del formulario para enviar sus datos al sistema
Precondicion	Que el usuario se haya registrado
Valor introducido	*Email* y contraseña
Resultado esperado	Coinciden los datos introducidos con los que hay en el sistema, se devuelve un token JWT que se almacena en *sessionStorage*
Resultado obtenido	Token JWT almacenado en *sessionStorage*
Resultado de la prueba	Correcto

Tabla 7.8: Prueba de caja negra 8. Inicio de sesión de usuario

PCN - 09	**Cerrar sesión**
Descripción	El usuario cierra su sesión.
Acciones realizadas	1. El usuario pulsa el botón ”Cerrar sesión” bajo la barra de navegación
Precondicion	Que el usuario haya iniciado sesión
Valor introducido	Ninguno
Resultado esperado	Se elimina el token JWT que se almacenaba en *sessionStorage*
Resultado obtenido	Token JWT eliminado en *sessionStorage*
Resultado de la prueba	Correcto

Tabla 7.9: Prueba de caja negra 9. Cerrar sesión

PCN - 10	**Modificación de la barra de sesión al iniciar sesión**
Descripción	Cuando se detecte un token en *sessionStorage* el aspecto de la barra de sesión cambia.
Acciones realizadas	1. Se introduce un token en *sessionStorage*
Precondicion	Ninguna
Valor introducido	Token JWT generado con los datos correspondientes
Resultado esperado	Cambio de los elementos en la barra de sesión, se ocultan los campos para iniciar sesión y registrarse y se pasa a mostrar un mensaje de bienvenida y un botón para cerrar sesión
Resultado obtenido	Modificación de los elementos de la barra de sesión
Resultado de la prueba	Correcto

Tabla 7.10: Prueba de caja negra 10. Modificación barra de sesión

PCN - 11	**Obtención de datos de un usuario**
Descripción	Se solicitan los datos de un usuario
Acciones realizadas	1. Se solicitan los datos del usuario
Precondicion	Ninguna
Valor introducido	Identificador del usuario
Resultado esperado	Se obtienen los datos del usuario, identificador, nombre, apellidos, *email*, página web, fecha de registro y rol
Resultado obtenido	Datos del usuario recibidos
Resultado de la prueba	Correcto

Tabla 7.11: Prueba de caja negra 11. Obtención de datos de usuario

PCN - 12	**Mostrar datos del usuario**
Descripción	Se muestran los datos de un usuario
Acciones realizadas	1. Se pulsa sobre el nombre del usuario
Precondicion	Ninguna
Valor introducido	Nada
Resultado esperado	Se muestran en la vista los datos del usuario, nombre, apellidos, *email*, página web y fecha de registro
Resultado obtenido	Datos del usuario mostrados
Resultado de la prueba	Correcto

Tabla 7.12: Prueba de caja negra 12. Mostrar datos de usuario

PCN - 13	**Mostrar datos del usuario que está identificado**
Descripción	Se muestran los datos del propio usuario identificado
Acciones realizadas	1. Se pulsa sobre el nombre del propio usuario en la tabla de experimentos
Precondicion	Haber iniciado sesión
Valor introducido	Nada
Resultado esperado	Se muestran en la vista los datos del usuario, nombre, apellidos, *email*, página web y fecha de registro junto con un botón para poder editar la página web
Resultado obtenido	Datos del usuario mostrados
Resultado de la prueba	Correcto

Tabla 7.13: Prueba de caja negra 13. Mostrar datos del usuario identificado

PCN - 14	**Modificar página web**
Descripción	Se modifica la página web del usuario identificado
Acciones realizadas	1. Se pulsa sobre el botón de editar página web en la vista de los datos del propio usuario 2. Se muestra un input para introducir la dirección 3. Se introduce la dirección y se pulsa el botón de editar
Precondicion	Estar identificado
Valor introducido	Dirección página web
Resultado esperado	Se muestra en la vista los datos del usuario junto con la dirección que se acaba de introducir
Resultado obtenido	Dirección web del usuario modificado
Resultado de la prueba	Correcto

Tabla 7.14: Prueba de caja negra 14. Modificar dirección web usuario

PCN - 15	**Obtener lista de instancias**
Descripción	Obtener una lista con todas las instancias
Acciones realizadas	1. Se pide una lista con todas las instancias
Precondicion	Ninguna
Valor introducido	Nada
Resultado esperado	Se obtiene una lista con las instancias del sistema
Resultado obtenido	Lista con las instancias del sistema
Resultado de la prueba	Correcto

Tabla 7.15: Prueba de caja negra 15. Obtener lista con las instancias

PCN - 16	**Mostrar datos de las instancias**
Descripción	Mostrar los datos de las instancias del sistema
Acciones realizadas	1. Se pulsa sobre el botón de la barra de navegación de ”Descarga de problemas” 2. Se cambia de vista
Precondicion	Ninguna
Valor introducido	Nada
Resultado esperado	Acceso a la vista con todas las instancias del sistema
Resultado obtenido	Vista de todas las instancias del sistema
Resultado de la prueba	Correcto

Tabla 7.16: Prueba de caja negra 16. Mostrar datos de las instancias del sistema

PCN - 17	**Obtener datos de una instancia**
Descripción	Obtener los datos de una instancia
Acciones realizadas	1. Se pide los datos de una instancia
Precondicion	Ninguna
Valor introducido	Identificador de la instancia a obtener
Resultado esperado	Se obtienen los datos de la instancia
Resultado obtenido	Datos de la instancia
Resultado de la prueba	Correcto

Tabla 7.17: Prueba de caja negra 17. Obtener una instancia

PCN - 18	**Mostrar una instancia**
Descripción	Mostrar los datos de una instancia
Acciones realizadas	1. En la vista que lista las instancias, se pulsa el botón de ”Ver problema” de una instancia
Precondicion	Ninguna
Valor introducido	Identificador de la instancia
Resultado esperado	Acceso a la vista de datos de una instancia
Resultado obtenido	Vista de datos de una instancia
Resultado de la prueba	Correcto

Tabla 7.18: Prueba de caja negra 18. Mostrar datos de una instancia

PCN - 19	**Mostrar botón eliminar instancia**
Descripción	Se muestra el botón de eliminar una instancia en la tabla
Acciones realizadas	1. Se accede a la vista que lista las instancias
Precondicion	Usuario identificado y con rol de profesor
Valor introducido	Ninguno
Resultado esperado	Se muestra junto con cada instancia una columna adicional con un botón para eliminar la instancia
Resultado obtenido	Se muestra el botón
Resultado de la prueba	Correcto

Tabla 7.19: Prueba de caja negra 19. Mostrar botón para eliminar instancia

PCN - 20	**Eliminar una instancia**
Descripción	El usuario elimina una instancia
Acciones realizadas	1. En la vista que lista las instancias, se pulsa el botón de ”Eliminar” de una instancia
Precondicion	Usuario identificado y con rol de profesor
Valor introducido	Ninguno
Resultado esperado	Se elimina la instancia del sistema
Resultado obtenido	Se ha eliminado la instancia del sistema
Resultado de la prueba	Correcto

Tabla 7.20: Prueba de caja negra 20. Eliminar una instancia

PCN - 21	**Añadir una instancia**
Descripción	El usuario añade una instancia
Acciones realizadas	1. En la barra de navegación, se pulsa ”Subir un problema” 2. Se rellenan los campos que se piden con los datos 3. Se pulsa el botón de ”Aceptar”
Precondicion	Usuario identificado
Valor introducido	Nombre de la instancia, lista de proyectos con el nombre de cada problema y tiempo de inicio de cada uno y la cantidad de recursos que se usarán de cada uno de los 4 posibles
Resultado esperado	Se añade la instancia al sistema
Resultado obtenido	Se ha añadido la instancia al sistema
Resultado de la prueba	Correcto

Tabla 7.21: Prueba de caja negra 21. Añadir una instancia

PCN - 22	**Obtener lista de experimentos**
Descripción	Obtener una lista con todos los experimentos
Acciones realizadas	1. Se pide una lista con todos los experimentos
Precondicion	Ninguna
Valor introducido	Nada
Resultado esperado	Se obtiene una lista con los experimentos del sistema
Resultado obtenido	Lista con los experimentos del sistema
Resultado de la prueba	Correcto

Tabla 7.22: Prueba de caja negra 22. Obtener lista con los experimentos

PCN - 23	**Mostrar datos de los experimentos**
Descripción	Mostrar los datos de los experimentos del sistema
Acciones realizadas	1. Se pulsa sobre el botón de la barra de navegación de ”Experimentos” 2. Se cambia de vista
Precondicion	Ninguna
Valor introducido	Nada
Resultado esperado	Acceso a la vista con todos los experimentos del sistema
Resultado obtenido	Vista de todos los experimentos del sistema
Resultado de la prueba	Correcto

Tabla 7.23: Prueba de caja negra 23. Mostrar datos de los experimentos del sistema

PCN - 24	**Mostrar botón añadir experimento**
Descripción	Se muestra el botón de añadir un experimento bajo la tabla
Acciones realizadas	1. Se accede a la vista que muestra los experimentos
Precondicion	Estar identificado
Valor introducido	Ninguno
Resultado esperado	Se muestra bajo la tabla de experimento con un botón para añadir un experimento
Resultado obtenido	Se muestra el botón
Resultado de la prueba	Correcto

Tabla 7.24: Prueba de caja negra 24. Mostrar botón para añadir experimento

PCN - 25	**Añadir un experimento**
Descripción	El usuario añade un experimento
Acciones realizadas	1. Se pulsa el botón de añadir experimento 2. Se muestra una nueva vista 3. Se rellenan los campos nombre, comentarios y DOI 4. Se pulsa el botón de ”Añadir experimento”
Precondicion	Usuario identificado
Valor introducido	Nombre del experimento, comentario y DOI
Resultado esperado	Se añade el experimento al sistema
Resultado obtenido	Se ha añadido el experimento al sistema
Resultado de la prueba	Correcto

Tabla 7.25: Prueba de caja negra 25. Añadir un experimento

PCN - 26	**Añadir un experimento sin DOI**
Descripción	El usuario añade un experimento sin el enlace al DOI
Acciones realizadas	1. Se pulsa el botón de añadir experimento 2. Se muestra una nueva vista 3. Se rellenan los campos nombre, comentarios y se deja el DOI vacío 4. Se pulsa el botón de ”Añadir experimento”
Precondicion	Usuario identificado
Valor introducido	Nombre del experimento, comentario
Resultado esperado	Se añade el experimento al sistema con el campo DOI vacío
Resultado obtenido	Se ha añadido el experimento al sistema con el campo DOI vacío
Resultado de la prueba	Correcto

Tabla 7.26: Prueba de caja negra 26. Añadir un experimento sin DOI

PCN - 27	**Eliminar un experimento**
Descripción	El usuario elimina un experimento
Acciones realizadas	1. En la vista que lista los experimentos, se pulsa el botón de ”Eliminar” de un experimento
Precondicion	Usuario identificado y con rol de profesor
Valor introducido	Ninguno
Resultado esperado	Se elimina el experimento del sistema
Resultado obtenido	Se ha eliminado el experimento del sistema
Resultado de la prueba	Correcto

Tabla 7.27: Prueba de caja negra 27. Eliminar un experimento

PCN - 28	**Mostrar botón eliminar experimento**
Descripción	Se muestra el botón de eliminar un experimento en la tabla
Acciones realizadas	1. Se accede a la vista que lista los experimentos
Precondicion	Usuario identificado y con rol de profesor
Valor introducido	Ninguno
Resultado esperado	Se muestra junto con cada experimento una columna adicional con un botón para eliminar el experimento
Resultado obtenido	Se muestra el botón
Resultado de la prueba	Correcto

Tabla 7.28: Prueba de caja negra 28. Mostrar botón para eliminar un experimento

PCN - 29	**Añadir experimento sin nombre**
Descripción	El usuario introduce todos los datos del experimento pero deja el campo del nombre vacío y pulsa el botón de ”Añadir experimento” para añadir los datos al sistema.
Acciones realizadas	1. El usuario pulsa el botón de ”Añadir experimento” bajo la tabla de experimentos 2. El usuario introduce sus datos en los campos dejando vacío el campo del nombre 3. El usuario pulsa el botón ”Añadir experimento” del formulario para enviar los datos al sistema
Precondicion	Usuario identificado
Valor introducido	Comentario y DOI
Resultado esperado	Se detecta que falta el valor del nombre, se rechaza la solicitud de añadir experimento y se muestra un mensaje de error
Resultado obtenido	Petición rechazada y se muestra el mensaje de que ha habido un problema
Resultado de la prueba	Correcto

Tabla 7.29: Prueba de caja negra 29. Añadir experimento sin nombre

PCN - 30	**Añadir experimento sin comentario**
Descripción	El usuario introduce todos los datos del experimento pero deja el campo del comentario vacío y pulsa el botón de ”Añadir experimento” para añadir los datos al sistema.
Acciones realizadas	1. El usuario pulsa el botón de ”Añadir experimento” bajo la tabla de experimentos 2. El usuario introduce sus datos en los campos dejando vacío el campo del comentario 3. El usuario pulsa el botón ”Añadir experimento” del formulario para enviar los datos al sistema
Precondicion	Usuario identificado
Valor introducido	Nombre y DOI
Resultado esperado	Se detecta que falta el valor del comentario, se rechaza la solicitud de añadir experimento y se muestra un mensaje de error
Resultado obtenido	Petición rechazada y se muestra el mensaje de que ha habido un problema
Resultado de la prueba	Correcto

Tabla 7.30: Prueba de caja negra 30. Añadir experimento sin comentario

Capítulo 8

Conclusiones

Desarrollado el servicio web de gestión y administración de instancias para los RCMPSP y su clasificación se puede concluir que:

- Se ha conseguido el objetivo principal del proyecto que era el desarrollo de un repositorio de instancias para los RCMPSP.
- Tras analizar detenidamente el servicio web final, se verifica que se han cumplido los requisitos funcionales de la aplicación definidos en la fase de análisis.
- Así mismo, se ha comprobado la correcta funcionalidad de los casos de uso siendo éstos verificados mediante la realización de pruebas al sistema, asegurándose que los resultados esperados de cada caso de uso son los esperados.
- Durante la fase de análisis se consideró la utilización de la metodología incremental para planificar el proyecto. Gracias a la utilización y la familiarización de esta metodología, la ejecución del proyecto se ha podido realizar siguiendo la planificación.
- También se pensó en la utilización de la herramienta Visual Studio Code y sus extensiones, lo que ha facilitado y agilizado el desarrollo del proyecto en la fase de implementación.

8.1 Valoración personal

He podido comprobar personalmente cómo se desarrolla un proyecto, desde las fases de análisis de objetivos y requisitos, pasando por el diseño y terminando con la implementación necesaria. También, me he dado cuenta de la importancia de tener una buena planificación inicial y un alcance fijado previamente para poder desarrollar el proyecto correctamente, porque aunque se ha alargado la duración de la fase de planificación se ha acortado la duración de la fase de implementación.

Haber elegido Trello como herramienta para tener controlado el desarrollo del proyecto ha sido de gran ayuda ya que, gracias a su formato, he podido controlar en todo momento el progreso de las tareas de un incremento.

Por último, ha servido como una gran fuente de experiencia y conocimientos sobre creación y gestión de una API-REST, comunicación segura con una base de datos y validación de peticiones realizadas. Así mismo, ha sido importante el aprendizaje sobre el desarrollo de una aplicación web segura y robusta gracias a las ayudas que ofrece Angular.

8.2 Trabajo futuro

A partir de los objetivos cumplidos para este proyecto y las necesidades y sugerencias que se han visto a lo largo de su desarrollo, se pueden identificar las mejoras que este servicio web puede tener:

- Como principal objetivo en un futuro trabajo se puede destacar la implementación de la funcionalidad necesaria para que los investigadores puedan proponer soluciones a las instancias que se encuentran en el sistema. Gracias a esto se puede tener un repositorio de instancias con la mejor solución alcanzada hasta el momento para las diferentes funciones objetivo y mediante las técnicas de resolución disponibles. Hacer accesible esta información a la comunidad investigadora permitirá un gran avance ya que no solo se dispondrá de un repositorio de las instancias de problemas RCMPSP sino que también estarán disponibles sus soluciones, estrategias con la que se han obtenido e información de los investigadores que las han desarrollado.
- Otra mejora es proporcionar más información sobre las instancias implementando la funcionalidad necesaria para que, utilizando la página web https://project.Rodrigomartin.dev se pueda mostrar a los investigadores un análisis completo de la instancia. Esta página muestra información muy útil para que un investigador pueda entender la complejidad de la instancia y, a partir de ésta, poder plantear una mejor solución.
- Como es el primer proyecto de esta escala que realiza el desarrollador, el desarrollo de algunos aspectos pueden hacerse de manera más óptima y con más claridad.
  - Se puede reordenar código para que sea más fácil entenderlo.
  - Limpiar variables que no sean realmente necesarias
  - Optimizar código. Hay funcionalidad que, implementada de otra manera, puede mejorar el tiempo de respuesta y la robustez de la aplicación.
  - Seguridad de datos. Al ser una aplicación que no está en fase de despliegue, hay información que puede ser sensible que no se ha cifrado al guardarla en el sistema o enviarla entre los servicios. Es una buena práctica cifrar por ejemplo las contraseñas de los usuarios para evitar que las roben fácilmente.
- Dada la importancia de utilizar protocolos seguros actualmente, sería interesante que la aplicación utilice HTTPS (*Hypertext Transfer Protocol Secure*) ya que a diferencia de HTTP (*Hypertext Transfer Protocol)*, éste permite no sólo cifrar el intercambio de información, sino autentificar de forma fehaciente al servidor, dando la garantía al usuario de que está subiendo o consultando la información en el sitio correcto y no es víctima de *phishing* u otro fraude similar.

Apéndices

Apéndice A

Manual de Instalación

Si se quiere instalar la aplicación web en una nueva máquina lo primero que se debe asegurar es que se esté utilizando una distribución Ubuntu de 64 bits. Se necesita instalar, si no lo tiene ya, Git para obtener el código fuente del proyecto. Para poder instalar Git se tienen que introducir los siguientes comandos en la terminal:

1. `$ sudo apt update`
2. `$ sudo apt install git`
3. `$ git --version`

Una vez comprobado que se tiene Git instalado, se procede a recoger el proyecto del repositorio, por lo que se tiene que ejecutar el siguiente comando:

```
$ git clone https://gitlab.inf.uva.es/oscpere/tfg_mpsplib
```

Como se van a ejecutar proyectos en contenedores Docker, se tiene que mover al directorio del proyecto, por lo que se ejecuta:

```
$ cd tfg_mpsplib
```

Ahí es donde se encuentra el archivo *docker-compose.yml* que es el encargado de crear y lanzar los contenedores, imágenes y volúmenes correspondientes. En una terminal se ejecutará el siguiente comando para lanzar la aplicación:

```
$ docker-compose up
```

Docker creará la base de datos, las imágenes de cada proyecto y sus contenedores, instalará las librerías y paquetes necesarios y lanzará toda la aplicación por lo que con esto ya se tendría la aplicación funcionando.

Para el funcionamiento de esta aplicación se han generado cuatro contenedores:

- Un contenedor llamado *app-flasko* que contendrá el proyecto Python con la API-REST lanzado en los puertos 8118:5555.
- Un contenedor llamado *midbadmin* que es un administrador de la base de datos lanzado en los puertos 5560:80.
- Un contenedor llamado *app-angular* que contendrá el proyecto Angular lanzado en los puertos 80:80.
- Un contenedor llamado *midbase* que contendrá la base de datos de la aplicación lanzado en los puertos 3306:3306.

Apéndice B

Manual de Usuario

B.1 Acceso público

Se empezará con la información a la que puede acceder cualquier usuario sin identificarse, la parte pública de la aplicación. Lo primero que deberá hacer el usuario es acceder a la dirección web http://virtual.lab.inf.uva.es:20092/. Estando la aplicación web en fase de desarrollo, se encuentra en un servidor de la universidad en el que sólo se puede acceder teniendo el equipo conectado a una red de la Universidad de Valladolid.

El usuario cuando accede al servicio web verá la página principal, figura 6.17. Tendrá la opción de, gracias a la barra de navegación superior, acceder a la vista de todas las instancias, del *ranking*, la vista que muestra los experimentos y la información de contacto con los profesores encargados de la gestión del servicio web.

Debajo de esta barra, habrá otra en la que el usuario puede identificarse o registrarse. Ya en el cuerpo de la interfaz, se tiene una breve presentación sobre el RCMPSP.

Si el usuario pulsa sobre el botón de ”Descarga de problemas”, accederá a una vista que le muestra todas las instancias que se encuentran en el sistema como se ve en la figura 6.18. En esta vista se muestran entre otros el nombre de la instancia, la cantidad de trabajos que tienen los proyectos que la componen, el número de proyectos que tiene la instancia, la cantidad de proyectos diferentes entre ellos, el número de tiempos de llegada diferentes y el número de recursos globales que tiene.

Además, se muestran dos columnas con la cantidad de soluciones que los usuarios han ido subiendo de cada instancia y un botón donde se podrá ver más información de la instancia.

El usuario podrá acceder a la lista de soluciones que tiene una instancia pulsando sobre el número de soluciones que tiene. Entonces será redirigido a otra vista que le mostrará primero la información de la instancia indicada en la vista anterior y debajo la lista de soluciones que los usuarios han subido junto con los datos de cada una, ver figura 6.24. Se mostrarán el usuario que la ha subido, la fecha en la que se añadió al sistema, las funciones objetivo que se buscan optimizar (APD, TMS y DPD) y el método usado, entre otros datos.

Si en cambio el usuario quiere ver la composición de la instancia, puede pulsar en el botón ”Ver problema” correspondiente a la instancia y será redirigido a una vista que le mostrará, utilizando etiquetas, la composición de ésta como en la vista 6.25. Se mostrarán datos como el nombre de la instancia, la lista de problemas que la componen junto con los tiempos de llegada de cada uno y la cantidad de cada recurso global que utiliza.

Al pulsar el botón ”Experimentos”, el usuario accederá a la vista de la figura 6.26 donde se le mostrarán todos los experimentos que los usuarios han subido al sistema. Se muestran el nombre, comentario, fecha de creación del experimento, quién lo ha subido y si tiene DOI asignado. A través de esta lista, si el usuario pulsa en el nombre de un usuario podrá ver sus datos. Esta vista incluirá el nombre y apellidos del usuario, el *email*, la fecha en la que se registró y la dirección de su página web si tiene, como se puede ver en la figura 6.31.

Por último, si el usuario quiere registrarse en la aplicación para poder acceder a la funcionalidad completa, tiene el botón ”Registrarse” bajo el logotipo de la Universidad de Valladolid. Si accede a esta vista podrá observar que se le pide el nombre, apellidos, *email*, contraseña y un enlace a su web. Si introduce todos los datos le saldrá un aviso en la pantalla confirmando el registro, como se puede ver en la figura 7.2, y podrá iniciar sesión y acceder al resto de opciones que, estando sin identificarse, no podría.

B.2 Acceso privado

B.2.1 Investigador

Un usuario investigador es cualquiera que se haya registrado en el sistema. Este usuario podrá acceder a todo lo que puede ver un usuario sin identificarse, pero a mayores se le dará la posibilidad de subir una nueva instancia, subir un nuevo experimento y de modificar datos de su usuario.

Cuando un usuario se identifica en el sistema, como se puede ver en la figura 6.18, aparecen dos nuevos botones en la barra de navegación. El primer botón lleva al usuario a la interfaz de subida de soluciones, figura 6.20, pero como aún esta funcionalidad no está implementada, no se permite que el usuario realice ninguna interacción.

El segundo botón que aparece es el de subir una instancia. Este lleva al usuario a la interfaz que gestiona la subida de nuevas instancias al sistema, figura 6.21. Lo primero que le pide al usuario será un nombre identificativo para la instancia. Después le pedirá que seleccione de entre los valores de actividades de un proyecto el tipo que utilizará para su instancia. Entonces se le pide que introduzca el número de proyectos que tendrá su instancia y se añadirá a la interfaz ese número de líneas para seleccionar el proyecto y el tiempo de inicio de cada uno. Por último, se le pedirá cuántos tipos de recursos querrá utilizar y tendrá que seleccionar un número entre el 1 y el 4. Entonces el usuario introducirá la cantidad de unidades que usará de cada uno de ellos y al darle al botón de ”Aceptar” saldrá un aviso en la parte superior diciendo que se ha añadido la instancia como en la figura 6.22 y se recarga la página.

Si un investigador accede a la vista de los experimentos podrá ver cómo se ha añadido un botón debajo de la tabla para poder subir un nuevo experimento. Si pulsa este botón, se le llevará a una interfaz, ver figura 7.25 que le permitirá introducir el nombre, comentario y, si tiene, DOI del experimento y publicarlo pulsando el botón de ”Añadir experimento”. Entonces, un aviso se mostrará en la parte superior de la página informando de que se ha creado con éxito, y se recargará la página.

B.2.2 Profesor

Un usuario profesor es el gestor del servicio web. Este usuario tendrá acceso a toda la funcionalidad de la aplicación web. Esto implica poder realizar toda la funcionalidad que se le permite utilizar a un usuario investigador pero, a mayores, se le permite eliminar instancias y experimentos.

Si un profesor entra en la vista que lista todas las instancias, en la tabla se añadirá una columna en la que aparecerá un botón para eliminar la instancia correspondiente si el profesor lo considera necesario como se ve en la figura 6.19.

De la misma manera, si el profesor entra en la vista que lista los experimentos, se añadirá una columna a la tabla en la que habrá un botón para eliminar el experimento correspondiente, figura 6.28.

Bibliografía

1. Rodríguez, P. Á.-C. *Simulador para la resolución de problemas de programación multiproyecto con restricción de recursos* https://uvadoc.uva.es/bitstream/handle/10324/49313/TFM-I-2083.pdf (2023).
2. González, F. B. *Nuevos métodos de resolución del problema de secuenciación de proyectos con recursos limitados* https://documat.unirioja.es/servlet/tesis?codigo=6857&info=resumen (2004).
3. Research, O. y Scheduling (OR&S), U. o. G. (*Operations Research & Scheduling Research Group* https://www.projectmanagement.ugent.be/ (2023).
4. Gantt, H. L. *Work wages and profit. The engineering magazine. Republished as Work, Wages and Profits* ISBN: 0879600489 (Hive Publishing Company, 1974).
5. Malcolm, D. G., Roseboom, J. H., Clark, C. E. y Fazar, W. Application of a Technique for Research and Development Program Evaluation. *Operations Research. Vol. 7(5).* http://dx.doi.org/10.1287/opre.7.5.646 (1959).
6. Kelley, J. y Walker, M. Critical-Path Planning and Scheduling. *Proceedings of the eastern joint computer conference.* https://dl.acm.org/doi/10.1145/1460299.1460318 (1959).
7. Fulkerson, D. R. A Network Flow Computation for Project Cost Curves. *Management science. Vol. 7 (2). Pp. 167-178.* https://dx.doi.org/10.1287/mnsc.7.2.167 (1961).
8. Steward, D. The Design Structure System: A Method for Managing the Design of Complex Systems. *IEEE Transactions on Engineering Management (28) 71-74.* http://dx.doi.org/10.1109/TEM.1981.6448589 (1981).
9. Baker, K. R. Introduction to Sequencing and Scheduling. *Wireless Sensor Network, Vol.2 (12).* (2011).
10. Bock, D. B. y Patterson, J. H. A Comparison of Due Date Setting, Resource Assignment and Job Preemption Heuristics for the Multiproject Scheduling Problem. *Decision Sciences, Vol. 21 387-402.* https://doi.org/10.1111/j.1540-5915.1990.tb01692.x (1990).
11. A. Sprecher R. Kolisch, A. D. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research, 80:94–102.* https://doi.org/10.1016/0377-2217(93)E0294-8 (1995).
12. Brucker, P. *Scheduling Algorithms* ISBN: 3540205241 (Springer, 2004).
13. I. Kurtulus, E. W. D. Multi-Project Scheduling: Categorization of Heuristic Rules Performance. *Management Science 28 (2):161-172.* https://doi.org/10.1287/mnsc.28.2.161 (1982).
14. Kolisch, R. y Padman, R. An integrated survey of deterministic project scheduling. *Omega Volume 29 (3).* https://doi.org/10.1016/S0305-0483(00)00046-3 (2001).
15. Kotwani, K., Yassine, A. y Zhao, Y. Scheduling resource constrained multi project DSM using modified simple GA and omeGA. *Working Paper, Dept. of IESE, UIUC* (2006).

16. Meier, C. y ans T. R. Browning, A. A. Y. Design Process Sequencing With Competent Genetic Algorithms. *Journal of Mechanical Design, 129(6): 566-585.* https://doi.org/10.1115/1.2717224 (2007).
17. Browning, T. R. y Yassine, A. A. Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics, 126 (2010) 212–228.* https://doi.org/10.1016/j.ijpe.2010.03.009 (2010).
18. Kolisch, R. y Sprecher, A. Resource-constrained multi-project scheduling: Priority rule performance revisited. *European Journal of Operational Research,Vol. 96 205–216.* https://doi.org/10.1016/S0377-2217(96)00170-1 (1997).
19. Kolisch, R., Schwindt, C. y Sprecher, A. Benchmark instances for project scheduling problems. *Project scheduling: recent models, algorithms and applications S. 197-212.* https://doi.org/10.1007/978-1-4615-5533-9_9 (1999).
20. Schwindt, C. ProGen/max: A new problem generator for different resource constrained project scheduling problems with minimal and maximal time lags. *Report WIOR 449. Institut für Wirtschaftstheorie und Operations Research, Universit at Karlsruhe.* (1995).
21. Homberger, J. A multi-agent system for the decentralized resource-constrained multi-project scheduling problem. *International Transactions in Operational Research 14(6), 565 - 589.* https://doi.org/10.1111/j.1475-3995.2007.00614.x (2007).
22. Eynde, R. y Vanhoucke, M. Resource-constrained multi-project scheduling: Benchmark datasets and decoupled scheduling. *Journal of Scheduling, Springer, vol. 23(3), 301-325.* https://doi.org/10.1007/s10951-020-00651-w (2020).
23. Aguilar, J. M. *¿Qué es el patrón MVC en programación y por qué es útil?* https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx (2023).
24. *Atlassian Trello* https://trello.com/ (2023).
25. *Overleaf, Online LaTex editor* https://es.overleaf.com/ (2023).
26. *LaTeX* https://www.latex-project.org/ (2023).
27. *Astah Professional* https://astah.net/products/astah-professional/ (2023).
28. *Unified Modeling Language (UML)* http://www.uml.org/ (2023).
29. *Visual Studio Code* https://code.visualstudio.com/ (2023).
30. *Draw.io (Diagrams for Confluence and Jira)* http://www.uml.org/ (2023).
31. *Angular* https://angular.io/ (2023).
32. *TypeScript* https://www.typescriptlang.org/ (2023).
33. *Javascript* https://www.javascript.com/ (2023).
34. *HTML* https://html.com/ (2023).
35. Cascading Style Sheets (CSS). https://www.w3.org/Style/CSS/Overview.en.html (2023).
36. *Bootstrap* https://getbootstrap.com/ (2023).
37. *Docker* https://www.docker.com/ (2023).
38. *Nginx* https://www.nginx.com/ (2023).
39. *Ubuntu* https://ubuntu.com/ (2023).
40. *MySQL* https://www.mysql.com/ (2023).
41. *SQLAlchemy* https://www.sqlalchemy.org/ (2023).
42. *Python* https://www.python.org/ (2023).
43. *Flask* https://flask.palletsprojects.com/ (2023).

44. *Postman* https://www.postman.com/ (2023).
45. *SQLAlchemy Relationship Configuration* https://docs.sqlalchemy.org/20/orm/relationships.html (2023).
46. *Flask-JWT* https://pythonhosted.org/Flask-JWT/ (2023).
47. *CLI Overview and Command Reference* https://angular.io/cli (2023).