ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

Trabajo Fin de Grado

Grado en Ingeniería Informática

APLICACIÓN WEB PARA LA PROGRAMACIÓN Y GESTIÓN DE PROYECTOS EN EL ÁMBITO DE LA FABRICACIÓN

Autor: Pablo Roldán Puebla

Directora: Dª. Eva Lucrecia Gibaja Galindo

Septiembre, 2025

UNIVERSIDAD D CÓRDOBA

Agradecimientos

En primer lugar, quiero agradecer a mi tutora Eva, por su siempre buena disposición y consejos, Desde que la conocí en primer año de carrera me quedó claro que con una profesora así de entregada a la educación sería increíble hacer mi trabajo fin de grado.

También a mi tutora de Prácticas de empresa Ana García, que desde que la conocí ha sido una persona que solo da todo de si misma para ayudar, enseñar y que me sintiera dentro de todo. Le agradezco haberme enseñado el tipo de ingeniero y persona que quiero llegar a ser.

Pero sobre todo a mi madre Isabel y mi padre Pablo, a mi hermana Cristina, al resto de mi familia (donde incluyo dentro a mi amiga Cristina) y a mis amigos, que me han acompañado, ayudado y comprendido a lo largo de este camino, que no ha sido fácil ni corto, pero ya se acerca a su fin.

APLICACIÓN WEB PARA LA PROGRAMACIÓN Y GESTIÓN EN EL ÁMBITO DE LA FABRICACIÓN

Pablo Roldán Puebla, Septiembre, 2025

Resumen

En este Trabajo de Fin de Grado se cubrirá el diseño e implementación de la arquitectura necesaria para una SPA *(Single Page Application)* destinada a la gestión y planificación de múltiples proyectos de montaje o fabricación de distintas máquinas agrícolas, visualizando dichos proyectos como tablas de Gantt. Creadas estas tablas conforme a los proyectos, se podrán tomar varias medidas de valor, como el camino crítico de las tareas del proyecto, cuándo se quedan libres los empleados necesarios para iniciar un proyecto, el coste en el tiempo de un proyecto, etc. Para todo esto se ha desarrollado un conjunto de productos software necesarios para dar cobertura al sistema, como una base de datos, una API RESTful que se ejecuta en el propio servidor, y la propia aplicación web, haciendo uso del *tech-stack MEAN (MongoDB, ExpressJS, Angular y NodeJS)*, para desplegar rápidamente una aplicación web dinámica y fácilmente escalable. Finalmente, se explorarán posibles líneas de mejora a llevar a cabo en el futuro de la aplicación.

Palabras clave: Aplicación web, Single Page Application, Gantt, Base de datos, API, RESTful, MEAN, MongoDB, ExpressJS, Angular, NodeJS, TypeScript.

El código fuente de este TFG y la documentación están disponibles en el repositorio oficial de github

WEB APPLICATION FOR PROJECT PLANNING AND MANAGEMENT IN THE FIELD OF FABRICATION

Pablo Roldán Puebla, September, 2025

Abstract

In this work we will cover the design and implementation of the necessary architecture for a SPA *(Single Page Application)* aimed at managing and planning multiple projects for the assembly or manufacturing of different machines, visualizing these projects as Gantt charts. Once these charts are created according to the projects, various valuable metrics can be derived, such as the critical path of the project tasks, when the required employees become available to start a project, the time cost of a project, scheduling new projects, etc. For all of this, a set of software products necessary to support the system has been developed, including a database, a RESTful API running on the server itself, and the web application itself, making use of the *MEAN (MongoDB, ExpressJS, Angular and NodeJS)* tech-stack to rapidly deploy a dynamic and easily scalable web application. Finally, possible lines of improvement to be carried out in the future of the application will be explored.

Key words: Web app, Single Page Application, Gantt, Database API, RESTful, MEAN, MongoDB, ExpressJS, Angular, NodeJS, TypeScript.

The source code for this work and the documentation are available on the official github repository

Índice de figuras

Índice de tablas

Índice

1. INTRODUCCIÓN

La idea de este proyecto fue inspirada por los campos en los que puede mejorar la empresa MORESIL SL [2], dedicada a la fabricación de maquinaria agrícola especializada.

Dentro de sus productos destacados, encontramos: cabezales para la recolección de cereales, máquinas para el cultivo del olivo, frutos secos y árboles frutales, y limpiadoras para seleccionar el grano. Esta maquinaria agrícola es producida en serie, generando varias unidades de un mismo modelo para sus diferentes clientes. La manufactura de la maquinaria es realizada por operarios especializados en diferentes áreas como mecánica o electricidad, en la mayoría de casos, varios operarios cooperan para realizar la misma tarea y producir el componente requerido, organizándose para producir el producto final para el cliente.

En las empresas dedicadas a la fabricación, ya sea de maquinaria o cualquier otro producto destinado a ser vendido directa o indirectamente a un cliente concreto, siempre intervienen varias fases para lograr crear el producto final. Estas fases pueden requerir el uso de maquinaria especializada,personal técnico o personas con conocimientos específicos en distintas áreas necesarias para el ensamblaje completo del producto. Ademas, tienen tanto un coste en tiempo como económico y de personal. Si un proyecto está mal gestionado o no se es capaz de calcular con una moderada precisión el coste total, existe la posibilidad de que se alargue más de lo esperado. Esto podría llevar al incumplimiento de plazos con los clientes o a costes inesperados por una mala gestión de los materiales. La tarea de planificar y adaptarse correctamente a los riesgos de un proyecto de fabricación ya es delicada. Si además añadimos la capa de complejidad de que una misma empresa lleva a cabo varios proyectos simultáneamente —ya sean idénticos (para el mismo producto) o diferentes (con distintos costes y necesidades)—, la administración de los recursos se vuelve aún más crítica. Una mala gestión puede provocar pérdidas de tiempo, parones inesperados y sobrecostes.

También sucede a menudo que, para una *PYME (Pequeña y/o Mediana Empresa)*, las soluciones profesionales de escritorio son costosas tanto de desarrollar como de adquirir, y muchas veces aún más complejas de administrar debido a la falta de personal técnico especializado. Esto puede obligar a destinar a una persona o equipo exclusivamente a la gestión de la aplicación. Una alternativa viable son las *soluciones web*, más accesibles y personalizables, como en el caso de la empresa Palfinger[3],una empresa que fabrica grúas hidráulicas, que obtuvo una mejora sustancial en su rendimiento por el uso de software de gestión de proyectos.

Estos son problemas comunes en la planificación de proyectos. De hecho, dentro de la familia de estándares ISO 21500, encontramos el estándar ISO 21502:2020 [4], que proporciona una guía internacional útil para organizaciones de cualquier tipo. Este estándar define un marco común de conceptos, términos y procesos que ayuda a gestionar proyectos sistemáticamente, asegurando el cumplimiento de los requisi-

tos esperados y adaptando buenas prácticas a cualquier tamaño o ámbito. Es natural que la gestión de proyectos haya atraído la atención de organismos reguladores. Como hemos visto, esta gestión responde a problemas específicos de cada empresa y, por tanto, aporta valor directamente, al permitir cumplir los objetivos empresariales. Esto se resume en la figura 1:

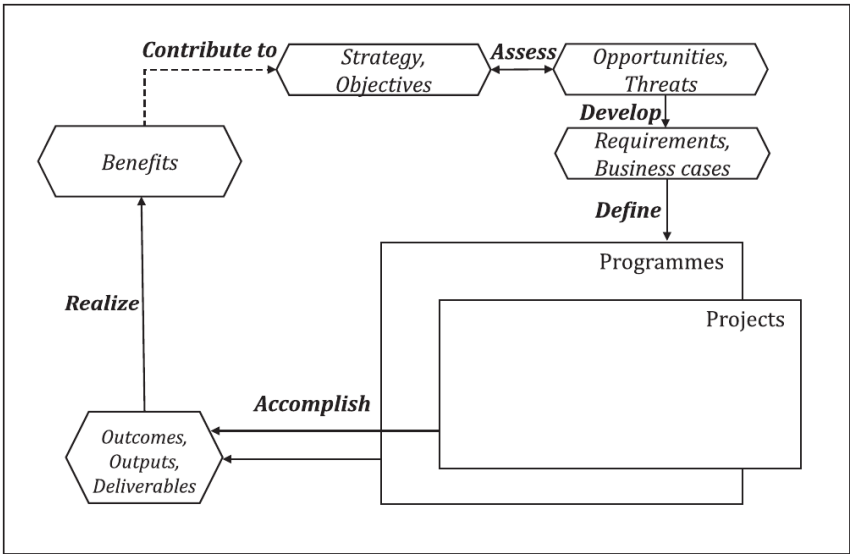

Figura 1: Cómo la gestión de proyectos aporta valor a la empresa[1]

En este trabajo se ha construido una aplicación web de ayuda a la planificación de proyectos en el ámbito de la fabricación de maquinaria agrícola. Realiza el seguimiento de dos recursos clave: el tiempo y el personal. Permitiendo el control de funcionalidades importantes para la Empresa MORESIL SL, representa un buen punto de partida para futuras ampliaciones. La aplicación permitirá crear proyectos mediante plantillas definidas por los usuarios, que podrán ser reutilizadas, facilitando el análisis de elementos como el *camino crítico del proyecto*, lo que permitirá predecir y visualizar los puntos mas delicados de los proyectos. Para facilitar esta gestión, la aplicación ofrecerá una interfaz intuitiva, simple y fácil de recordar.

2. DEFINICIÓN DEL PROBLEMA

2.1. Definición del problema real

En el entorno de la fabricación de maquinaria agrícola, las pequeñas y medianas empresas como MORESIL SL afrontan retos importantes al intentar planificar varios proyectos de montaje a la vez. La existencia de distintos encargos simultáneos requiere coordinar máquinas, operarios y plazos con gran precisión, algo difícil de lograr cuando se cuenta únicamente con herramientas genéricas que no consideran jornadas laborales, el camino critico del proyecto, las holguras de las tareas ni dependencias específicas de producción. Además, las soluciones profesionales disponibles suelen requerir licencias costosas y formación especializada, por lo que muchas pymes acaban recurriendo a hojas de cálculo y asumiendo el riesgo de errores manuales. Estas circunstancias generan solapamientos de tareas y demoras imprevistas que acaban reduciendo la competitividad de la empresa.

2.2. Definición del problema técnico

Para dar respuesta a estos problemas, el sistema debe permitir la definición de plantillas de proyecto reutilizables —con tareas, duraciones y dependencias—, la asignación automática de recursos según su disponibilidad, el cálculo dinámico del camino crítico y la holgura de cada tarea y la propagación de retrasos tanto dentro de un mismo proyecto como los proyectos sucesores, todo ello a través de una interfaz web ágil que evite recargas de página. Los pormenores de estas funcionalidades se presentan en las *secciones de Diseño y Arquitectura*.

2.2.1. Funcionamiento

La solución se articula en tres capas: la capa cliente, responsable de la interacción con el usuario; la capa de servidor, que expone la API REST con sus validaciones, manejo de errores y se ejecutan tareas periódicas de actualización de retrasos y holguras; y la capa de base de datos, donde MongoDB guarda la información . Cada una de estas capas se describe en detalle en la sección *Arquitectura del Sistema*.

2.2.2. Entorno

Para llevar a cabo este proyecto se ha definido un entorno de trabajo ágil y reproducible: en el cliente basta con un navegador moderno con soporte ES6; en el

servidor se emplea Node.js (v20 o superior) con Express 5.x y MongoDB 8.x; y en el desarrollo se utilizan VS Code, Git y GitHub, junto con npm *(node packet manager)* para la gestión de dependencias.

2.2.3. Vida esperada

Se prevé que la aplicación sea plenamente operativa durante al menos doce meses sin necesidad de cambios sustanciales. Tras ese periodo, los cambios en los procesos de fabricación, las nuevas versiones de librerías o la aparición de nuevos requisitos (por ejemplo, integración con otros sistemas) podrían suponer actualizaciones o ampliaciones de la plataforma.

2.2.4. Ciclo de mantenimiento

No se ha planteado un plan de mantenimiento más allá de la actualización periódica de librerías y la corrección de errores menores, dado que no contaremos con retroalimentación sobre las necesidades futuras de MORESIL SL.

2.2.5. Competencia

Las soluciones disponibles actualmente en el mercado (como Microsoft Project Web, Smartsheet, Zoho Projects, Basecamp o Asana) ofrecen funcionalidades de planificación y colaboración, pero no están diseñadas para producción industrial ni combinan en un mismo entorno plantillas reutilizables, asignación automática de recursos y gestión dinámica de retrasos. El análisis detallado de estas alternativas se presenta en la sección *Antecedentes*.

2.2.6. Aspecto externo

La interfaz ha sido diseñada para maximizar la claridad y la eficiencia. Cuenta con un menú siempre accesible en la vista principal, ventanas flotantes que no interrumpen el flujo de trabajo y notificaciones de éxito o error tras cada operación.

2.2.7. Estandarización

El desarrollo respetará los estándares web actuales (HTML5, CSS3, JavaScript moderno) y por los principios REST en la API.

2.2.8. Calidad y fiabilidad

Siguiendo patrones de buen diseño (Arquitectura en capas, patrón DAO/DTO), la arquitectura se separa claramente en capas y valida los datos en cada nivel, La integridad de datos será asegurada frente a caídas. Además, se realizará un conjunto de pruebas (unitarias y funcionales) cuyos resultados se describen en la sección *Pruebas y errores*.

2.2.9. Programa de tareas

Con respecto a la programación original del anteproyecto, no hubo demasiado cambio, a excepción de la implementación del sistema, la cual aumentó de forma imprevista su duración debido a factores cambiantes en la empresa que obligaron a redefinir algunos requisitos básicos del sistema a desarrollar. Aunque estos cambios no surtieron un efecto demasiado grande en el tiempo tareas como el diseño, probaron ser un factor que añadió complejidad al desarrollo.

Estos efectos pueden verse en la tabla1.

Semana	Tarea
1	Búsqueda de Módulos útiles para la implementación del software
2-3	Análisis y definición de requisitos
4-6	Diseño del sistema que se va a implementar
7-14	Desarrollo e implementación del sistema
15	Pruebas del software
16-17	Reunificación de la documentación realizadas en cada fase

Tabla 1: Distribución temporal del proyecto

2.2.10. Pruebas

Para este proyecto, los escenarios críticos cubrirán el acceso y autenticación, la gestión de plantillas y proyectos, la asignación automática de recursos, la visualización en calendario y Gantt, y el proceso periódico de actualización de retrasos.

2.2.11. Seguridad

En todas las comunicaciones entre cliente se aplica una sanitación de las entradas en el servidor para prevenir inyecciones, además de autenticar la validez de las solicitudes realizadas mediante tokens de seguridad de acceso a los servicios de la API.

3. OBJETIVOS

El objetivo general de este proyecto será el desarrollo de una aplicación web para la gestión y programación de proyectos de fabricación de maquinaria así como la gestión de los recursos humanos necesarios para la ejecución de las fases del proyecto. La aplicación se ceñirá a los a los estándares de la metodología UWE [5] de desarrollo. Para alcanzar este objetivo principal, han sido planteados los siguientes objetivos específicos:

1. **Gestión del personal involucrado:**
   - Permitir el seguimiento de la disponibilidad del personal involucrado en los proyectos.
   - Posibilitar la introducción de personal mediante un listado informatizado para facilitar la gestión de la información.
   - Ofrecer una vista detallada de la carga de trabajo de cada empleado.
2. **Creación y almacenamiento de plantillas de proyectos:**
   - Desarrollar una funcionalidad que permita la creación de plantillas reutilizables para los proyectos.
   - Garantizar el almacenamiento seguro de las plantillas para su posterior reutilización.
   - Facilitar la adaptación de las plantillas para ajustarlas a fechas específicas
   - Permitir la edición de plantillas por parte de usuarios.
3. **Visualización de la línea temporal de proyectos:**
   - Implementar una herramienta visual para monitorear la línea temporal de todos los proyectos en curso.
   - Evaluar si los plazos de los proyectos se mantienen dentro de los límites aceptables mediante técnicas tales como el camino crítico y el calculo de holguras en las tareas.
   - Analizar el impacto de cada proyecto en los demás dentro del cronograma general.
4. **Visualización simultánea de proyectos:**
   - Desarrollar una interfaz que permita la visualización simultánea de múltiples proyectos.

Manteniendo el objetivo general de este TFG, durante su desarrollo ha sido necesario priorizar y reducir los objetivos específicos. Esto se debe a que la escala inicial resultó demasiado amplia para el tiempo establecido para el desarrollo de un TFG. En la siguiente tabla se recogen los objetivos específicos incluidos en el anteproyecto que no han podido ser abordados.

Objetivo	Descripción
1	Integración de recursos empresariales (conjunto de requisitos completos)
3	Gestión de presupuestos (conjunto de requisitos completos)
5.4	Permitir la simulación de escenarios para optimizar la programación de proyectos.
6.2	Facilitar la comparación entre proyectos en términos de tiempos, costos y recursos empleados.
6.3	Implementar filtros y vistas personalizables para mejorar la experiencia del usuario.

Tabla 2: Objetivos específicos del anteproyecto que no han sido considerados

4. ANTECEDENTES

En el mercado existen diversas herramientas web para la gestión de proyectos en ámbitos como el marketing y la ingeniería, pero pocas enfocadas específicamente en la fabricación, ademas, para el volumen de negocio de la fabricación (de maquinaria agrícola o cualquier manufactura de este tipo) hace que el conjunto de usuarios (y por tanto, licencias) sea alto, aumentando el coste de usar estas plataformas. Es fundamental analizar cómo estas soluciones abordan los objetivos generales y específicos planteados en este proyecto.

Una de las primeras soluciones ampliamente adoptadas fue **Basecamp** (2004), centrada en la organización básica de tareas y la comunicación entre miembros del equipo. Su simplicidad favoreció su uso en proyectos pequeños, pero no incluye funciones para la planificación productiva ni para la gestión de recursos industriales [6].

Basecamp solo ofrece un plan pro plus de 299 euros mensuales por el plan ilimitado, y una opción mas modesta de 15 euros por mes por usuario

Microsoft Project Web Access (2007), como parte del ecosistema de Project Server, permite una planificación más robusta, incluyendo asignación de tareas y seguimiento de recursos. Aunque fue adoptado en entornos corporativos, su orientación sigue siendo generalista y no contempla las particularidades del entorno de fabricación [7]. Ofrece varios planes de pago, pero todos manteniendo la forma de pago por usuario. En el plan mas barato el coste de cada usuario son 9,40 euros mensuales, en planes superiores el precio se dispara hasta 28.10 euros y 51,50 euros por usuario al mes.

Asana (2011) introdujo distintas vistas de tareas y automatización de procesos, lo que facilitó la gestión de múltiples proyectos y flujos de trabajo. Aunque es eficaz en contextos colaborativos, no incorpora herramientas específicas para la producción ni para la gestión de recursos físicos [8]. En su plan mas barato, el pago asciende a 140 euros (aproximadamente) por usuario anualmente, planes superiores con mas funcionalidades ascienden a 316 euros por usuario anuales.

JIRA Advanced Roadmaps (2015), antes *Portfolio*, añadió la posibilidad de visualizar múltiples proyectos en una única línea temporal, facilitando la detección de dependencias y conflictos de planificación. A pesar de este avance, su diseño sigue enfocado al desarrollo de software, sin integrar variables propias de entornos industriales [9].

La única forma de acceder a este producto es pagando suscripción a *Jira Cloud*, con el precio ascendiendo a 11,70 euros mensuales por usuario.

Smartsheet (2021) [10] permite gestionar tareas, presupuestos y cronogramas mediante una interfaz intuitiva. Es especialmente útil para visualizar la secuencia de tareas dentro de un proyecto y ha sido utilizado en sectores como la construcción, un caso destacado se el de la empresa **Palfinger** [3], que aumentó su rendimiento con esta herramienta. Sin embargo, no ofrece integración con recursos empresariales ni permite evaluar el impacto de un proyecto sobre otros. Tampoco incluye una visión unificada de todos los proyectos ni contempla funciones específicas para la fabricación, como planificación de recursos de producción o sincronización con inventarios.

Esta alternativa ofrece planes *Pro* y *Business*, con un coste de 7.76 y 16.38 euros al mes por usuario, aunque la forma de pago que puede ser mas atractiva es el plan *Enterprise*, cuyo precio está abierto a negociación.

Zoho Projects (2006) ofrece diagramas de Gantt y funcionalidades como *baseline* y *critical path*, que permiten detectar desviaciones entre lo planificado y lo ejecutado [11].Zoho Projects es una opción rentable y fácil de usar[12], lo que la hace atractiva para equipos que buscan una curva de aprendizaje accesible. Pero Desde la perspectiva de los objetivos específicos de este trabajo fin de grado , no ofrece reutilización estructurada de proyectos ni visualización simultánea de varios en una línea de tiempo integrada. Tampoco permite establecer relaciones entre proyectos conectados. Al igual que las anteriores, no fue diseñada para la industria de fabricación de maquinaria.

Esta es la única alternativa que ofrece un plan gratuito, aunque solo para 3 usuarios máximo por organización. para acomodarlo al volumen de trabajo de la empresa es necesario pagar suscripción a sus planes *Premium* o *Enterprise*, los cuales cuestan 4,31 euros y 8,62 euros por usuario al mes.

Ninguna de estas herramientas responde de forma completa a los requisitos específicos de este proyecto. No se ha identificado una solución que permita gestionar proyectos de fabricación con una visión global y simultánea, además algunas de las alternativas no ofrecen calculo y/o visualización del camino crítico. Esto justifica el desarrollo de una alternativa que incluya la reutilización de plantillas y una visualización interactiva del estado de los proyectos en curso. Una herramienta centrada en la fabricación permitiría mejorar la planificación, optimizar el uso de recursos y

reducir costes, aumentando así la competitividad en el entorno industrial.

5. RECURSOS

5.1. Recursos Humanos

- Pablo Roldán Puebla: Realizador del proyecto.
- Eva Lucrecia Gibaja Galindo: Directora del TFG

5.2. Recursos Software

- Angular 19.0 y módulos útiles para la implementación: framework para desarrollar el cliente de la SPA[13].
- TypeScript: lenguaje fuertemente tipado usado por Angular[14].
- NodeJS: software para el backend en el lado del servidor[15].
- ExpressJS: framework de NodeJS para la generación de APIs RESTful[16].
- MongoDB: Base de datos no SQL usada en la aplicación web[17].
- Visual Studio Code: IDE usado para desarrollar el proyecto.
- Apache HTTP server: para ejecutar la aplicación web.
- Overleaf: Para la gestión de documentos del proyecto.
- Git y GitHub: para gestionar las versiones de la aplicación.
- Draw.io: Software de edición de diagramas usado en el Diseño de la aplicación.

5.3. Recursos Hardware

Dispositivo	**Memoria**	**Procesador**	**Núcleos**	**Sistema Operativo**
Lenovo Thinkpad T590	16GB RAM DDR4 UDIMM NO-ECC	Intel i5-8565U	4 núcleos físicos 8 núcleos lógicos	Dual Boot: Windows 11 Pro Manjaro Linux Xahea

Tabla 3: Recursos Hardware del proyecto

6. ANÁLISIS DEL PROBLEMA

En esta sección se expondrán los requisitos mas importantes para el diseño de la aplicación, obtenidos de entrevistas con los clientes y métodos para la obtención de requisitos (tormentas de ideas, prototipos en papel, cuestionarios, etc), además de los objetivos mas relevantes previamente citados y alcanzables dentro del plazo de tiempo disponible para el proyecto.

Estos requisitos son los que han sido usados como entrada a las primeras fases del diseño del producto web.

6.1. Requisitos funcionales

RF1. El sistema deberá permitir la creación de plantillas de proyectos.

RF2. El sistema deberá permitir modificar plantillas de proyectos y guardar cambios.

RF3. El sistema deberá permitir añadir tareas a un proyecto.

RF4. El sistema deberá permitir guardar plantillas para su uso futuro.

RF5. El sistema deberá permitir visualizar proyectos individuales.

RF6. El sistema deberá permitir dentro de un proyecto ver las características de sus tareas.

RF7. El sistema deberá permitir visualizar los proyectos en una línea de tiempo global o un calendario.

RF8. El sistema deberá permitir ver el avance de las tareas de los proyectos.

RF9. El sistema deberá permitir comprobar cómo los adelantos o retrasos de un proyecto influye en el resto de proyectos de la línea temporal.

RF10. El sistema deberá permitir la identificación de usuarios mediante un identificador único de usuario.

RF11. El sistema debe diferenciar entre administradores y operarios.

RF12. El sistema deberá permitir a los administradores crear usuarios.

RF13. El sistema deberá permitir marcar la disponibilidad de un operario para realizar o asignar una tarea en un proyecto.

RF14. El sistema debe permitir asignar usuarios a tareas.

RF15. El sistema debe permitir crear y editar proyectos a realizar.

RF16. El sistema debe ofrecer información actualizada sobre el estado de los proyectos los proyectos.

RF17. El sistema debe mantener la sesión de un administrador durante un día.

RF18. El sistema permitirá visualizar la holgura y camino crítico de las tareas de cada proyecto.

6.2. Requisitos no funcionales

RNF1. El sistema se ejecutará en un entorno web para asegurar su correcto funcionamiento en un amplio espectro de equipos.

RNF2. El sistema deberá ser realizado con Angular por su buen desempeño en aplicaciones que son intensas en el lado del usuario.

RNF3. El sistema usará bases de datos no SQL como MongoDB por la facilidad de la implementación y alto desempeño en operaciones *CRUD*.

RNF4. El sistema implementará una interfaz cómoda y sencilla que garantice su facilidad de uso en equipos de escritorio.

RNF5. El servidor de la aplicación se ejecutará en Apache debido a su disponibilidad dentro de la empresa.

RNF6. Las plantillas de proyectos serán almacenadas en una base de datos.

RNF7. Las tareas y enlaces sólo existirán si existe su proyecto o plantilla.

6.3. Requisitos de información

RINF1. Cada tarea en el sistema consistirá en la realización de una operación sobre una pieza.

RINF2. El sistema deberá gestionar la relación de dependencias entre tareas, capturando información sobre qué tareas deben completarse antes de asignar nuevas.

RINF3. El sistema deberá almacenar información sobre las tareas asignadas a un usuario.

6.4. Especificación de casos de uso

6.4.1. Especificación de actores

El primer paso para el análisis de los requisitos mediante casos de uso será especificar los actores que participarán en la aplicación.

Tras un acercamiento inicial a los requisitos, se han encontrado los actores descritos a continuación

Actor	Descripción
Usuario sin sesión	Actor que no ha iniciado sesión en la aplicación, su principal tarea es iniciar sesión con unas credenciales válidas en caso de tenerlas
Administrador	Actor principal de la aplicación, representa a un usuario con credenciales válidas que ha entrado en la aplicación , por tanto encargado de desencadenar la funcionalidad principal.
Tiempo	Actor que se encuentra principalmente en segundo plano dentro la aplicación, encargado de gestionar la actualización de datos de los proyectos en relación a sus restricciones temporales.

Tabla 4: Actores del sistema

6.4.2. Diagrama de casos de uso

Los requisitos mínimos establecidos en el apartado anterior servirán de guía para modelar los flujos de acciones que puede realizar un usuario.

Teniendo esto en cuenta, nos queda el diagrama de la figura2 que nos proporciona una vista de los casos de uso ajustándose a los problemas a resolver dentro del dominio de la aplicación, teniendo en cuenta que estos casos de uso están diseñados para satisfacer los requisitos de la aplicación.

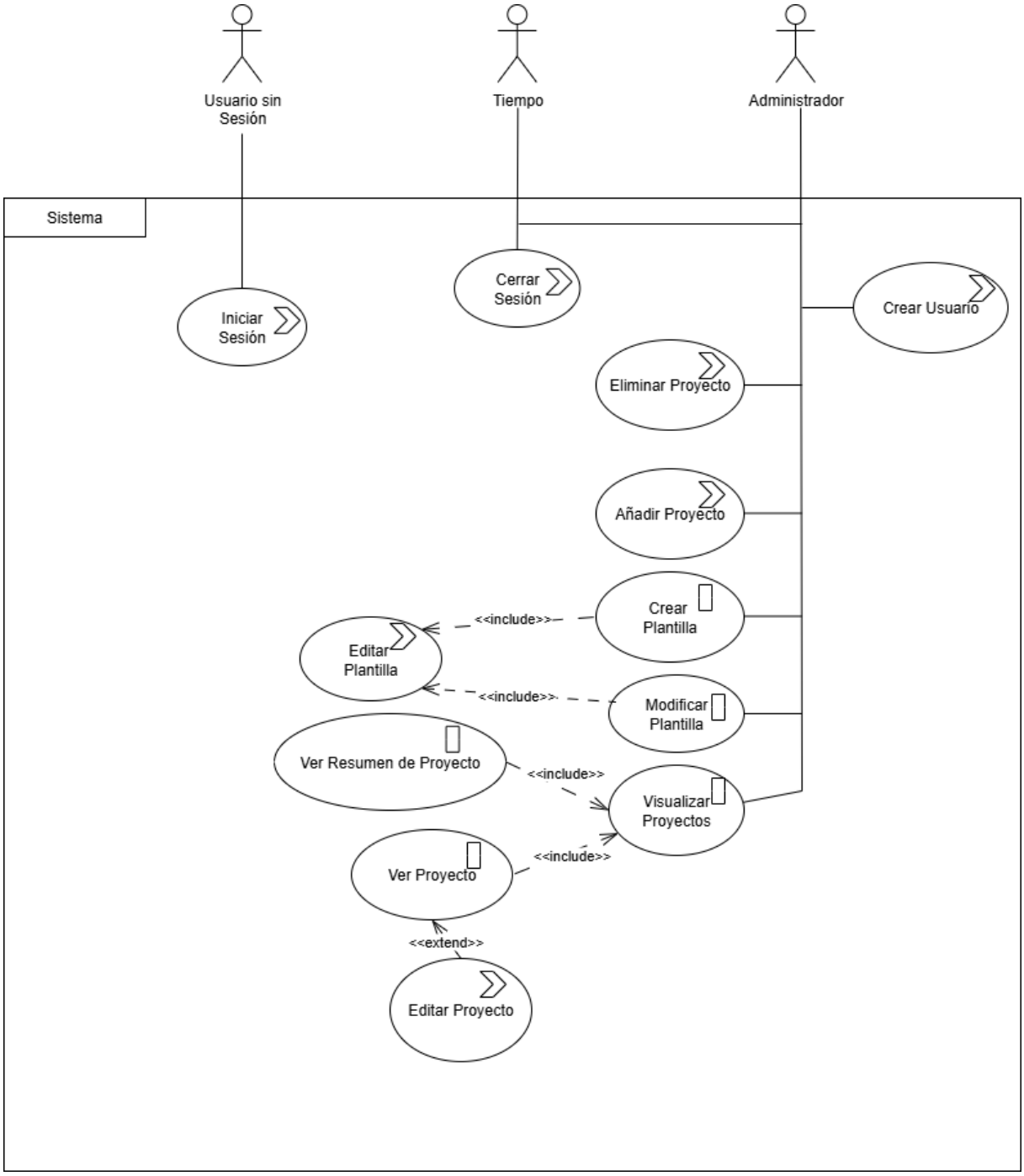

Figura 2: Diagrama de Casos de uso

6.4.3. Caso de uso - Iniciar sesión

Nombre	Iniciar sesión
Identificador	CU-01
Descripción	Permite al usuario iniciar sesión en el sistema.
Actores	Usuario sin Sesión
Precondiciones	El usuario no debe tener ninguna sesión activa
Postcondiciones	Usuario autenticado y sesión de usuario creada
Flujo Principal	1. El caso de uso comienza cuando el usuario desea iniciar sesión. 2. El sistema pide al usuario las credenciales de inicio de sesión. 3. El usuario introduce un nombre de usuario y contraseña válidos. 4. El sistema verifica credenciales.
Flujos Alternativos	▪ El usuario introduce datos incorrectos: 3(a). se muestra mensaje de error. ▪ Fallo en el servidor: 3(b). El sistema no verifica nada hasta que vuelva a haber conexión.

Tabla 5: Caso de uso - Iniciar sesión

6.4.4. Caso de uso - Cerrar sesión

Nombre	Cerrar sesión
Identificador	CU-02
Descripción	Permite al usuario iniciar sesión en el sistema.
Actores	Administrador, Tiempo
Precondiciones	El usuario debe tener una sesión iniciada
Postcondiciones	Sesión en la aplicación cerrada.
Flujo Principal	1. El caso de uso comienza cuando el usuario desea cerrar sesión. 2. El usuario elige cerrar la sesión. 3. El sistema pide confirmación del usuario. 4. El usuario confirma el cierre de sesión.
Flujos Alternativos	▪ El tiempo de sesión expiró: 1(a). El sistema cierra la sesión.

Tabla 6: Caso de uso - Cerrar sesión

6.4.5. Caso de uso - Crear usuario

Nombre	Crear usuario
Identificador	CU-03
Descripción	Permite a crear usuarios dentro de la aplicación
Actores	Administrador
Precondiciones	El usuario debe tener una sesión iniciada
Postcondiciones	Nuevo usuario creado dentro de la aplicación.
Flujo Principal	1. El caso de uso comienza cuando el usuario accede a la opción para crear usuario. 2. El sistema pide las credenciales del nuevo usuario. 3. El Administrador introduce *a*) Nombre del nuevo usuario. *b*) Contraseña del nuevo usuario. *c*) Tipo de usuario. 4. El sistema comprueba las credenciales. 5. El sistema crea al nuevo usuario.
Flujos Alternativos	▪ El Administrador introdujo credenciales inválidas: 5(a). El sistema notifica al Administrador del error.

Tabla 7: Caso de uso - Crear usuario

6.4.6. Caso de uso - Crear plantilla

Nombre	Crear plantilla
Identificador	CU-04
Descripción	Permite a crear plantillas de proyectos dentro de la aplicación
Actores	Administrador
Precondiciones	El usuario debe tener una sesión iniciada
Postcondiciones	Nueva plantilla de proyecto creada dentro de la aplicación.
Flujo Principal	1. El caso de uso comienza cuando el usuario accede a la opción para crear plantilla. 2. El sistema pide al usuario introducir el nombre de la nueva plantilla. 3. El Administrador introduce el nombre de la plantilla. 4. include «Editar Plantilla».
Flujos Alternativos	▪ proceso cancelado: cualquier punto. El Administrador ha cancelado manualmente el proceso.

Tabla 8: Caso de uso - Crear plantilla

6.4.7. Caso de uso - Modificar plantilla

Nombre	Modificar plantilla
Identificador	CU-05
Descripción	Permite a modificar plantillas de proyectos dentro de la aplicación
Actores	Administrador
Precondiciones	El usuario debe tener una sesión iniciada y debe existir una plantilla
Postcondiciones	plantilla de proyecto modificada creada dentro de la aplicación.
Flujo Principal	1. El caso de uso comienza cuando el usuario accede a la opción para modificar plantilla. 2. El sistema muestra las plantillas que existen dentro de la aplicación. 3. El Administrador selecciona una plantilla. 4. «include» Editar plantilla.
Flujos Alternativos	▪ proceso cancelado: cualquier punto. El Administrador ha cancelado manualmente el proceso.

Tabla 9: Caso de uso - Modificar plantilla

6.4.8. Caso de uso - Editar plantilla

Nombre	Editar plantilla
Identificador	CU-06
Descripción	Permite editar una plantilla de proyecto en la aplicación
Actores	Administrador
Precondiciones	El usuario debe tener una sesión iniciada y debe existir una plantilla
Postcondiciones	plantilla de proyecto editada dentro de la aplicación.
Flujo Principal	1. El caso de uso comienza cuando el usuario accede a la edición de una plantilla. 2. El sistema muestra las tareas de la plantilla con sus correspondientes holguras. 3. Mientras no haya acabado de editar una plantilla: *a*) El Administrador selecciona una de las tareas. *b*) El sistema muestra los datos de la tarea. *c*) El Administrador cambia los datos de la tarea. *d*) El sistema calcula el camino crítico de la plantilla. *e*) El sistema muestra la nueva tarea modificada. 4. el Administrador elige guardar los cambios realizados. 5. el sistema guarda los nuevos datos.
Flujos Alternativos	▪ proceso cancelado: cualquier punto. El Administrador ha cancelado manualmente el proceso. ▪ nueva tarea: 3.a(a) El administrador añade una nueva tarea.

Tabla 10: Caso de uso - Editar plantilla

6.4.9. Caso de uso - Añadir proyecto

Nombre	Añadir proyecto
Identificador	CU-07
Descripción	Permite a Añadir un nuevo proyecto en la aplicación
Actores	Administrador
Precondiciones	El usuario debe tener una sesión iniciada y debe existir una plantilla
Postcondiciones	proyecto creado dentro de la aplicación.
Flujo Principal	1. El caso de uso comienza cuando el usuario accede a la creación de un nuevo proyecto. 2. El sistema muestra las plantillas que existen dentro de la aplicación. 3. El Administrador selecciona una plantilla. 4. El Administrador introduce la fecha para el inicio del proyecto. 5. El sistema comprueba la disponibilidad de los operarios. 6. El sistema asigna tareas a los operarios disponibles. 7. El sistema muestra al Administrador el nuevo proyecto.
Flujos Alternativos	▪ proceso cancelado: cualquier punto. El Administrador ha cancelado manualmente el proceso. ▪ no hay operarios suficientes: 6(a). El sistema no ha encontrado suficientes operarios para el proyecto en la fecha asignada. El caso de uso finaliza.

Tabla 11: Caso de uso - Añadir proyecto

6.4.10. Caso de uso - Ver proyecto

Nombre	Ver proyecto
Identificador	CU-08
Descripción	Permite ver un proyecto en la aplicación
Actores	Administrador
Precondiciones	El usuario debe tener una sesión iniciada y debe existir un proyecto
Postcondiciones	visualizado proyecto dentro de la aplicación.
Flujo Principal	1. «include» Ver proyectos. 2. El Administrador selecciona un proyecto. 3. El sistema muestra al usuario las tareas del proyecto. 4. El sistema muestra el camino crítico de las tareas del proyecto.
Flujos Alternativos	▪ Proceso cancelado: cualquier punto. El Administrador ha cancelado manualmente el proceso. ▪ editar proyecto: a partir del paso 4. «extend» Editar proyecto.

Tabla 12: Caso de uso - Ver proyecto

6.4.11. Caso de uso - Editar proyecto

Nombre	Editar proyecto
Identificador	CU-09
Descripción	Permite editar un proyecto en la aplicación
Actores	Administrador
Precondiciones	El usuario debe tener una sesión iniciada y debe estar visualizando un proyecto
Postcondiciones	Editado proyecto dentro de la aplicación.
Flujo Principal	1. El caso de uso comienza cuando el usuario accede a un proyecto. 2. Mientras no haya acabado de editar un proyecto: *a*) El Administrador selecciona una de las tareas. *b*) El sistema muestra los datos de la tarea. *c*) El Administrador cambia los datos de la tarea. *d*) El sistema calcula y propaga la holgura de la tarea. *e*) El sistema muestra la nueva tarea modificada. 3. el Administrador elige guardar los cambios realizados. 4. el sistema guarda los nuevos datos.
Flujos Alternativos	▪ Proceso cancelado: cualquier punto. El Administrador ha cancelado manualmente el proceso. ▪ Error de guardado: 4(a). El sistema notifica al usuario sobre el error al guardar los nuevos datos.

Tabla 13: Caso de uso - Editar proyecto

6.4.12. Caso de uso - Ver resumen de proyecto

Nombre	Ver resumen de proyecto
Identificador	CU-10
Descripción	Permite ver un resumen del estado del proyecto en la aplicación
Actores	Administrador
Precondiciones	El usuario debe tener una sesión iniciada
Postcondiciones	Editado proyecto dentro de la aplicación.
Flujo Principal	1. «include» Ver proyectos. 2. El caso de uso comienza cuando El Administrador visualiza los proyectos activos en un día concreto. 3. El Administrador selecciona un proyecto. 4. El sistema muestra los datos de interés al usuario.
Flujos Alternativos	▪ Proceso cancelado: cualquier punto. El Administrador ha cancelado manualmente el proceso.

Tabla 14: Caso de uso - Ver resumen de proyecto

6.4.13. Caso de uso - Ver proyectos

Nombre	Ver proyectos
Identificador	CU-11
Descripción	Permite ver un resumen del estado del proyecto en la aplicación
Actores	Administrador
Precondiciones	El usuario debe tener una sesión iniciada
Postcondiciones	Visualizados proyectos dentro de la aplicación.
Flujo Principal	1. El caso de uso comienza cuando El Administrador entra a la aplicación con una sesión iniciada. 2. El sistema muestra al usuario los proyectos de la aplicación. 3. El sistema muestra los datos de interés al usuario.
Flujos Alternativos	▪ No se pudieron obtener los datos: 3(a). El sistema notifica al usuario que no se pudieron obtener los datos solicitados.

Tabla 15: Caso de uso - Ver proyectos

6.4.14. Caso de uso - Eliminar proyecto

Nombre	Eliminar proyecto
Identificador	CU-12
Descripción	Permite ver un resumen del estado del proyecto en la aplicación
Actores	Administrador
Precondiciones	El usuario debe tener una sesión iniciada
Postcondiciones	Eliminado proyecto dentro de la aplicación.
Flujo Principal	1. El caso de uso comienza cuando El Administrador selecciona un proyecto para borrado. 2. El sistema pide confirmación al usuario. 3. El usuario confirma la operación 4. El sistema elimina el proyecto de la aplicación.
Flujos Alternativos	▪ Error al borrar la tarea: 4(a). El sistema notifica al usuario el error al borrar la tarea.

Tabla 16: Caso de uso - Eliminar proyecto

6.4.15. Matriz de trazabilidad de casos de uso

Con todos los casos uso especificados, se procede a comprobar el grado en el que satisfacen los requisitos funcionales de la aplicación mediante la matriz de trazabilidad, expuesta en la tabla17.

CU	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**	**R12**	**R13**	**R14**	**R15**	**R16**	**R17**	**R18**
CU1										X								
CU2										X							X	
CU3											X	X						
CU4	X			X														
CU5		X		X														
CU6		X		X														X
CU7			X										X	X	X			
CU8					X	X												
CU9						X									X			X
CU10						X									X			
CU11																X		
CU12							X	X	X							X		

Tabla 17: Matriz de trazabilidad de casos de uso

6.5. Diagramas de actividad

El modelado del flujo de acciones de cada caso de uso se realizará mediante los de diagramas de actividad.

Estos diagramas serán sencillos, describiendo los pasos que seguirá un usuario para realizar uno de los casos de uso dentro del dominio de la aplicación.

Con estos diagramas se representa gráficamente el flujo de acciones que se realizarán para llevar a cabo un caso de uso paso por paso, por lo que para entender bien estos diagramas solo hay que prestar atención a la especificación del caso de uso al que representa cada uno.

Aunque no se requiere un grado de detalle elevado, podremos tener una idea inicial del comportamiento dinámico de la aplicación.

6.5.1. Diagrama de Actividad - Iniciar Sesión

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Iniciar sesión*5.

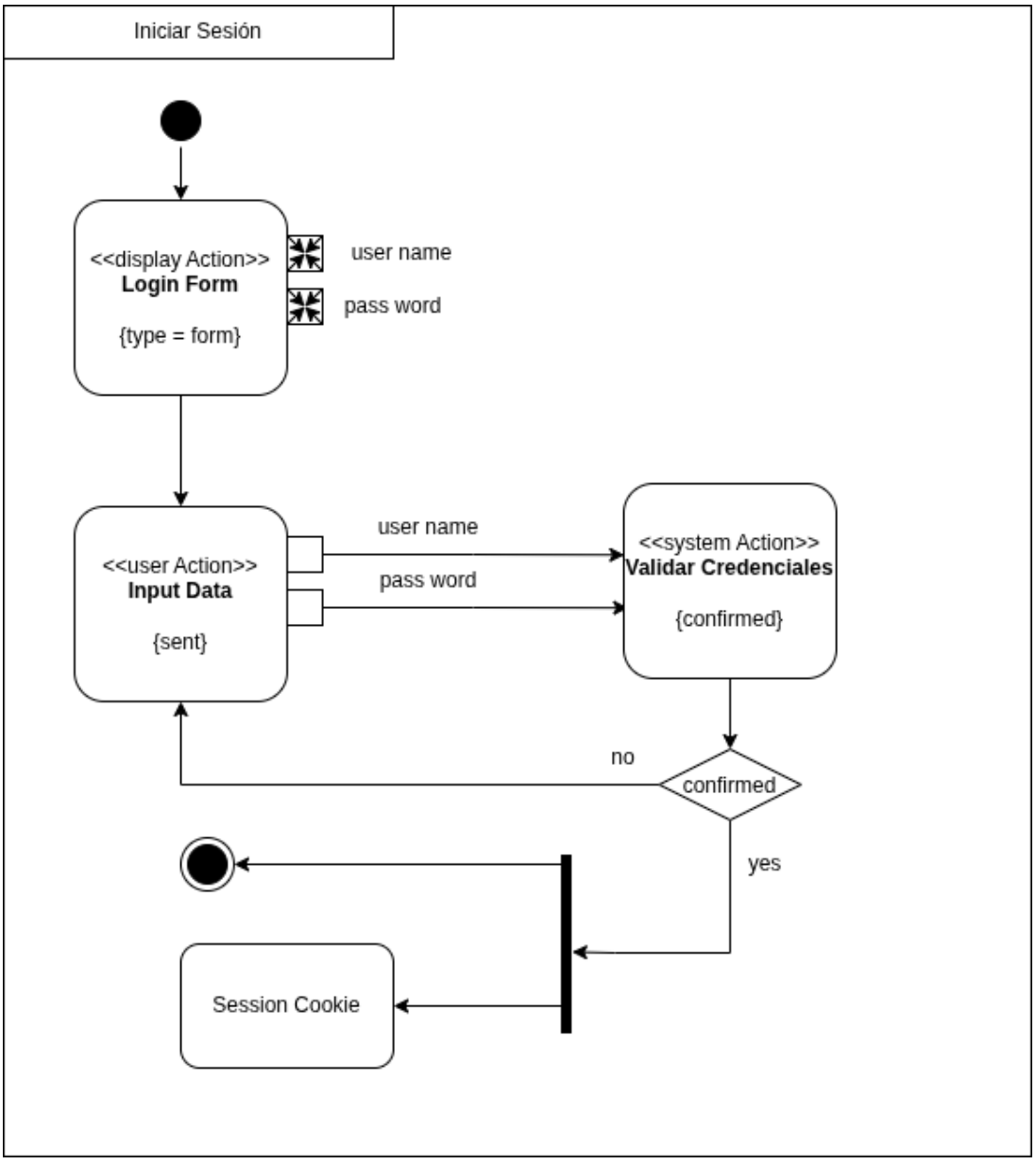

Figura 3: Diagrama de Actividad - Iniciar Sesión

6.5.2. Diagrama de Actividad - Cerrar Sesión

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Cerrar sesión*6.

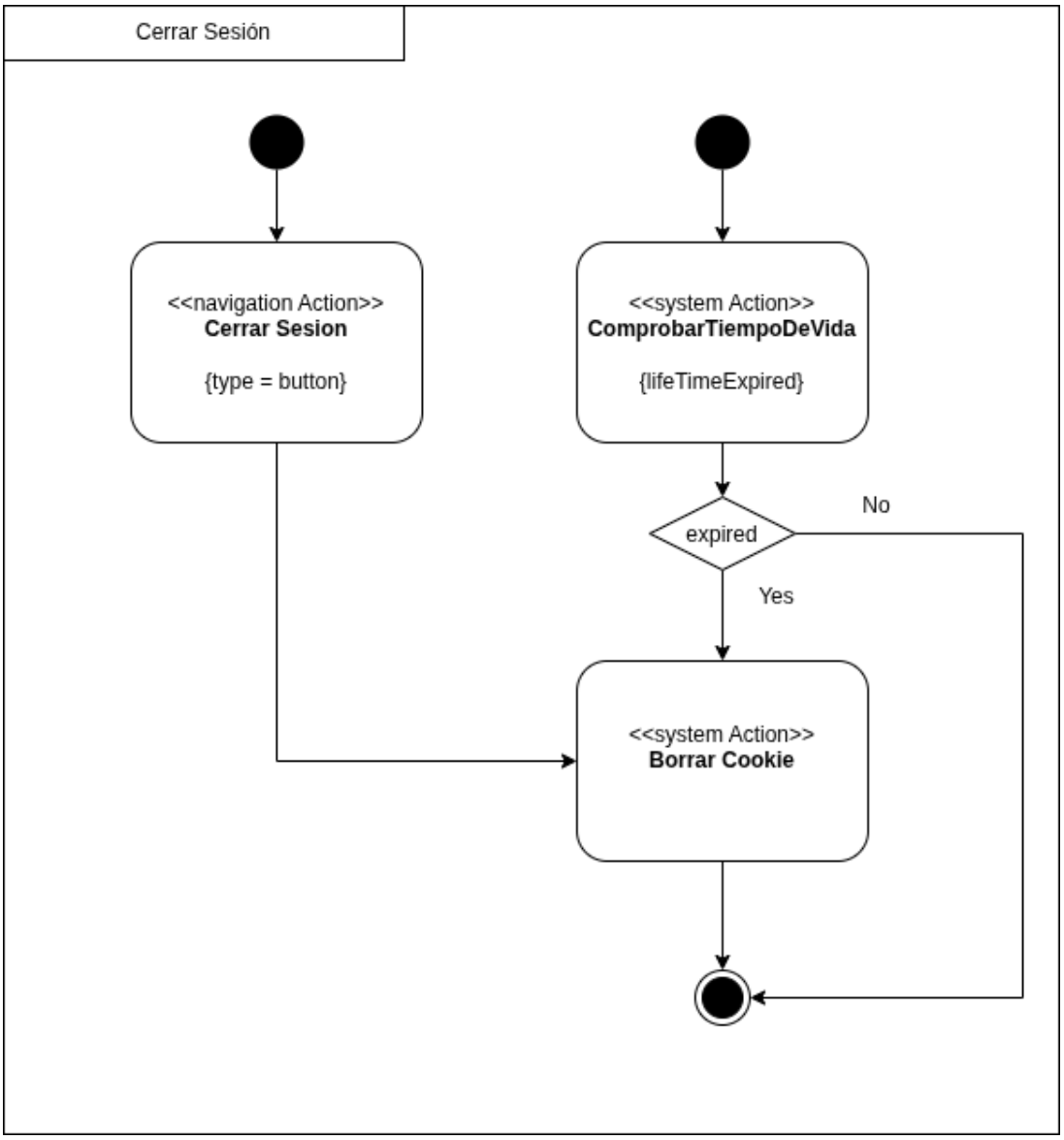

Figura 4: Diagrama de Actividad - Cerrar Sesión

6.5.3. Diagrama de Actividad - Crear Usuario

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Crear usuario*7.

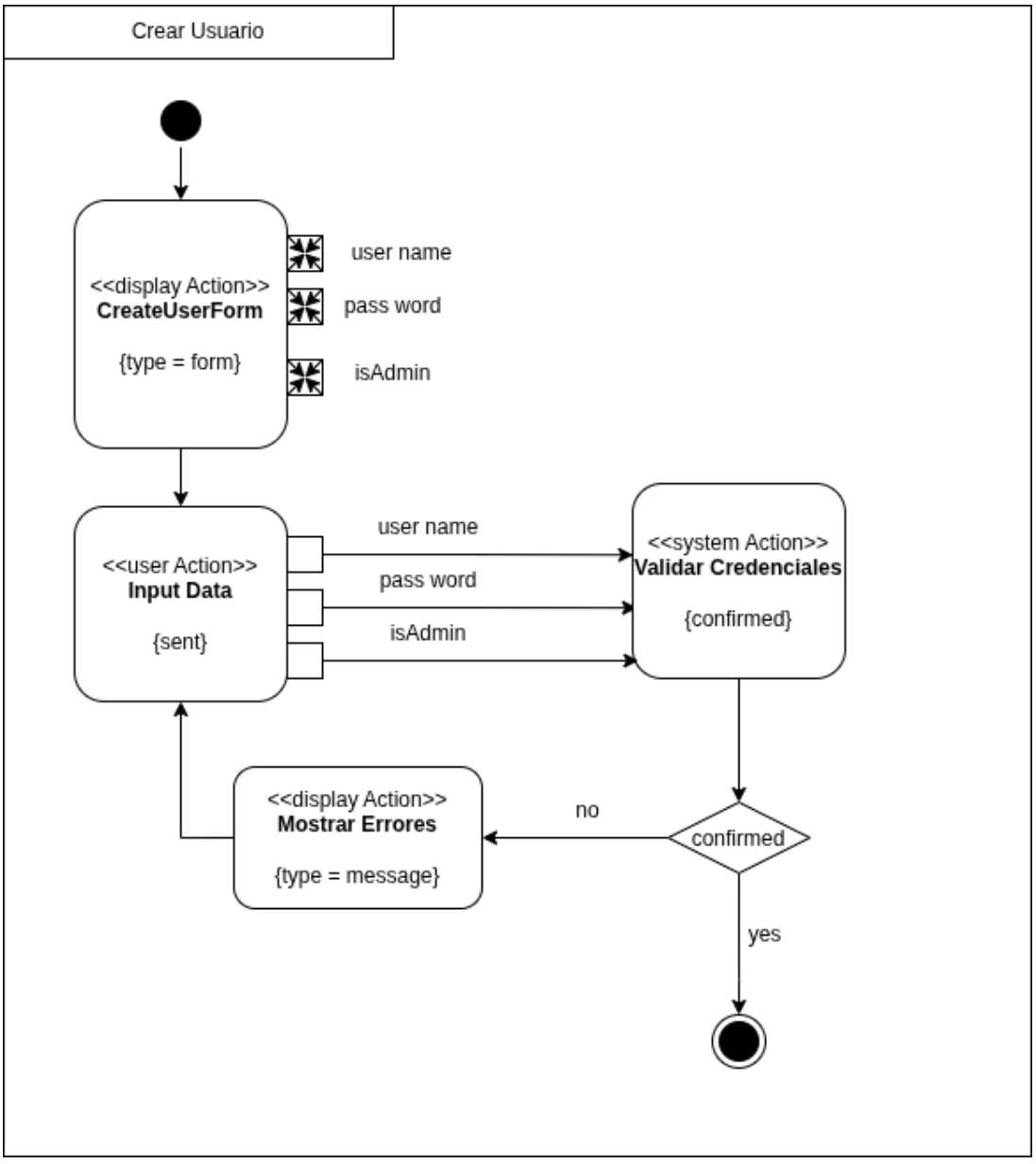

Figura 5: Diagrama de Actividad - Crear Usuario

6.5.4. Diagrama de Actividad - Crear Plantilla

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Crear plantilla*8.

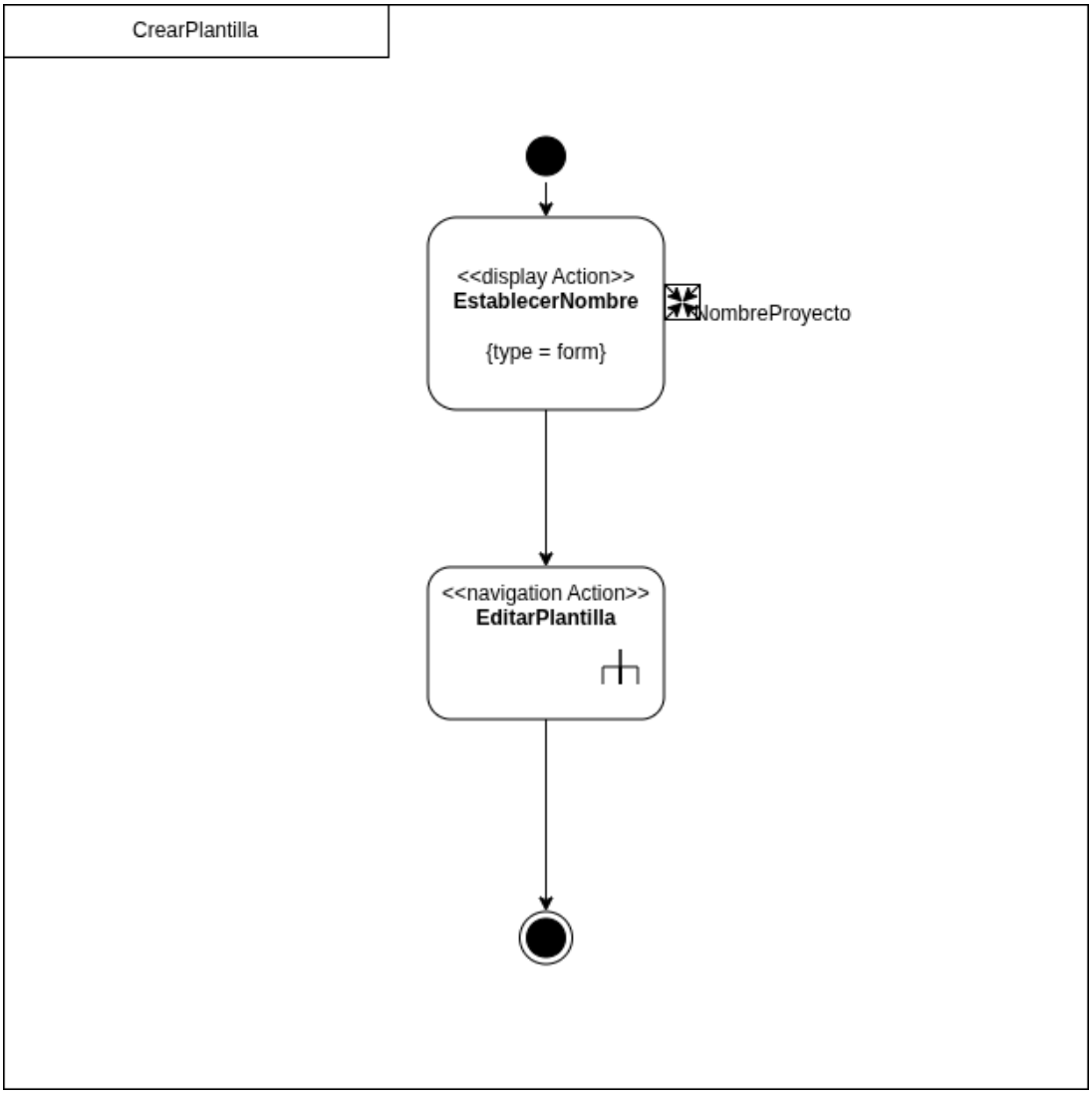

Figura 6: Diagrama de Actividad - Crear Plantilla

6.5.5. Diagrama de Actividad - Modificar Plantilla

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Modificar plantilla*9.

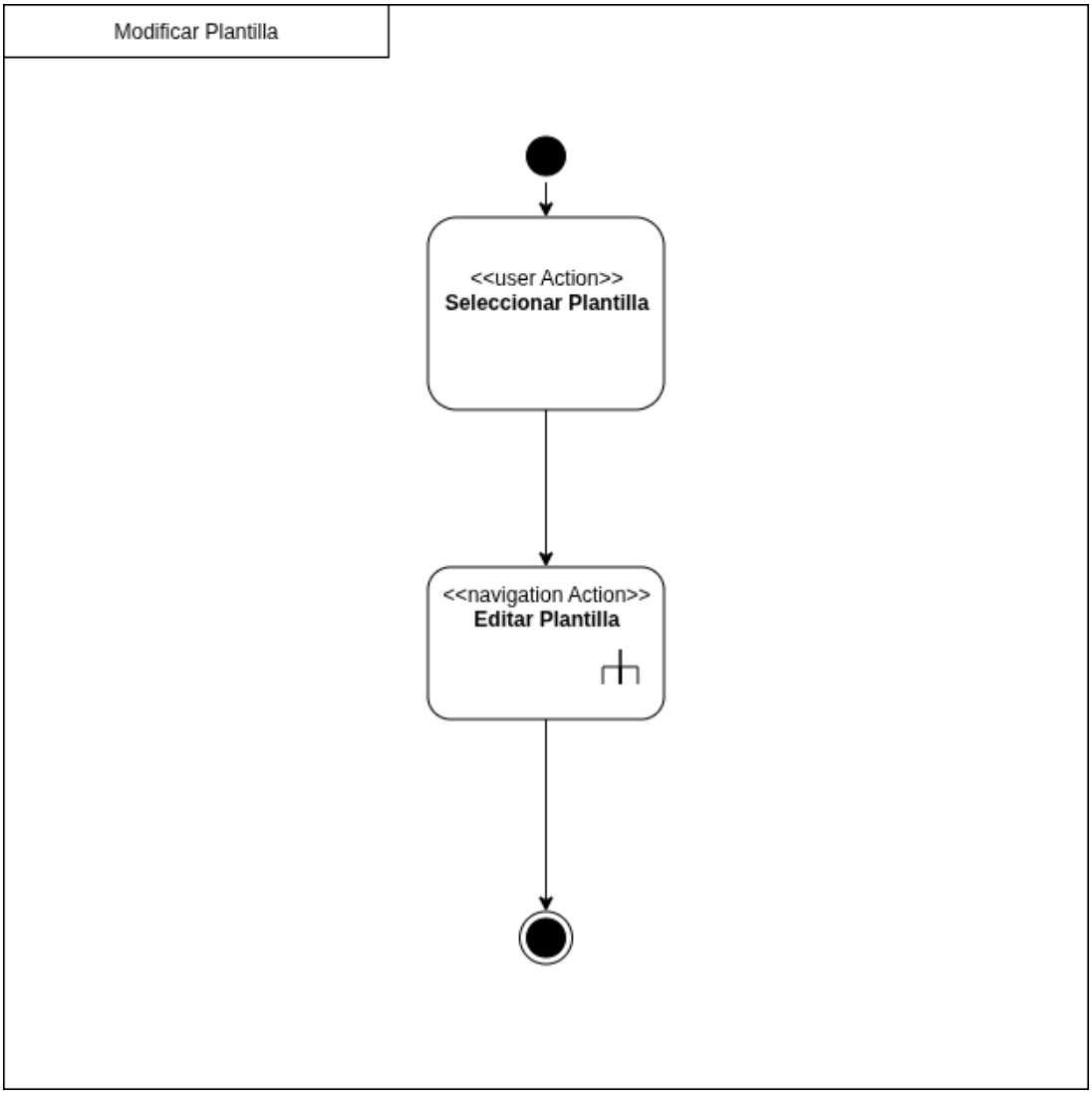

Figura 7: Diagrama de Actividad - Modificar Plantilla

6.5.6. Diagrama de Actividad - Editar Plantilla

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Editar plantilla*10.

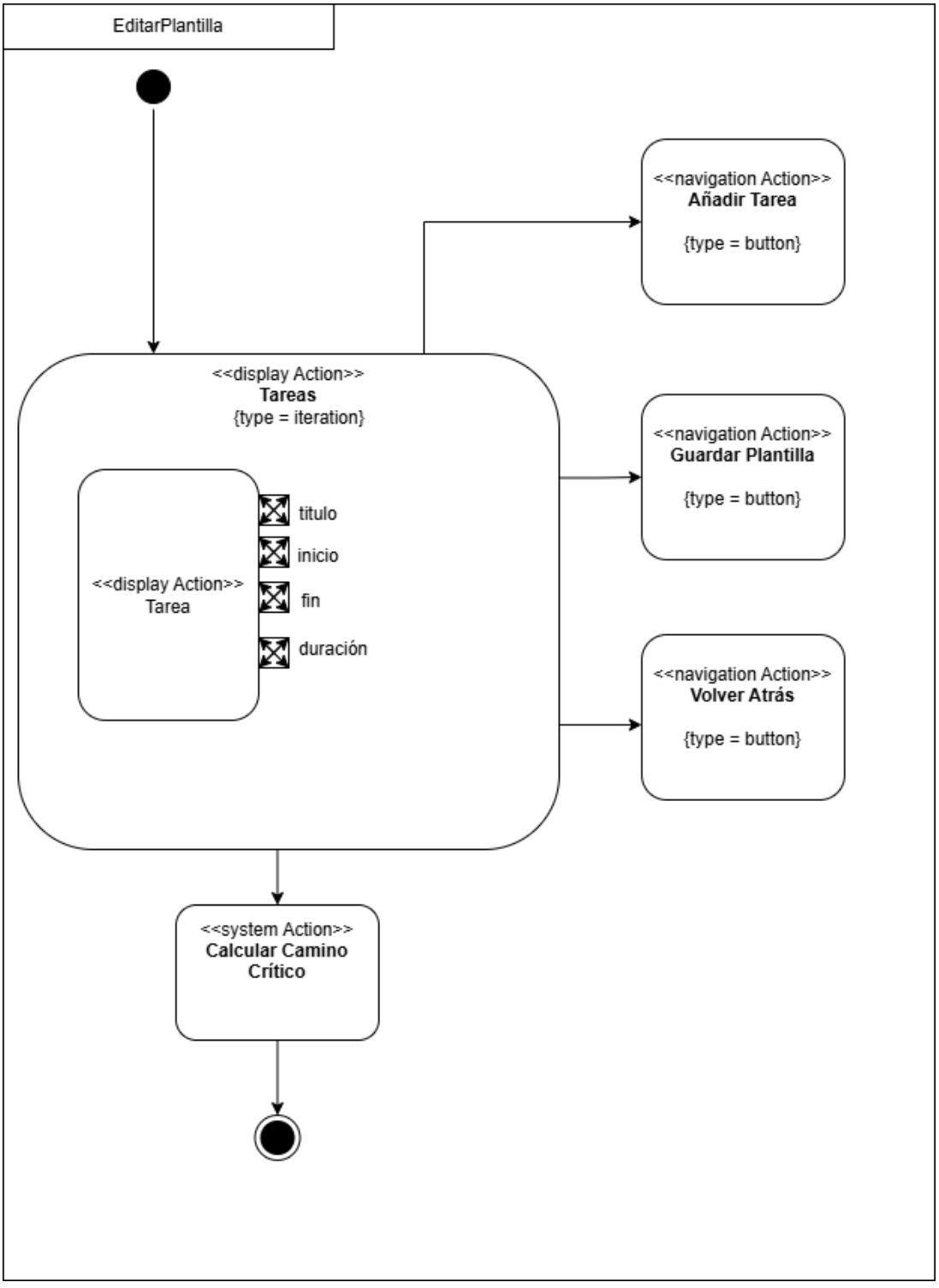

Figura 8: Diagrama de Actividad - Editar Plantilla

6.5.7. Diagrama de Actividad - Crear Proyecto

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Añadir proyecto*11.

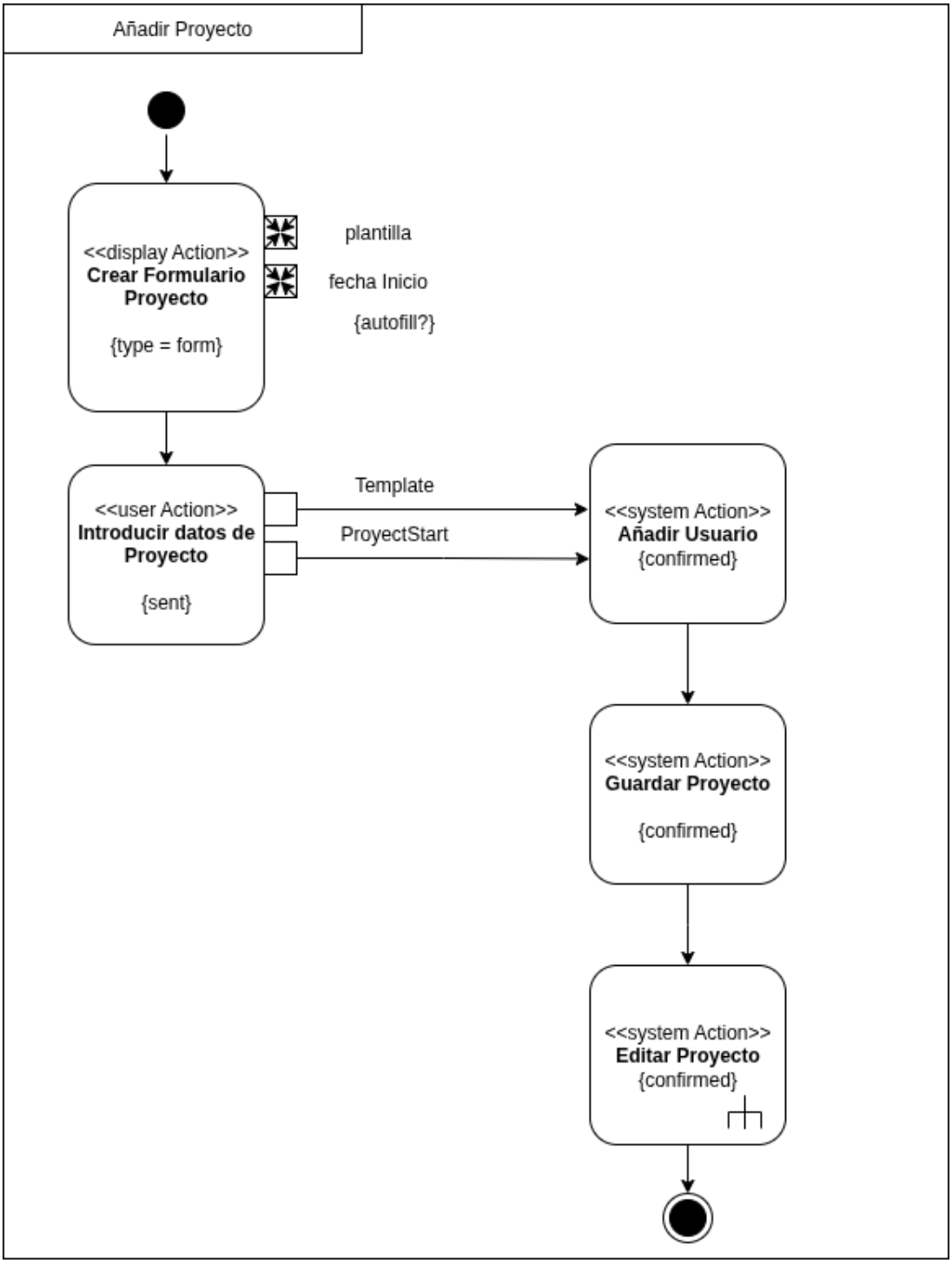

Figura 9: Diagrama de Actividad - Añadir Proyecto

6.5.8. Diagrama de Actividad - Ver Proyecto

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Ver proyecto*12.

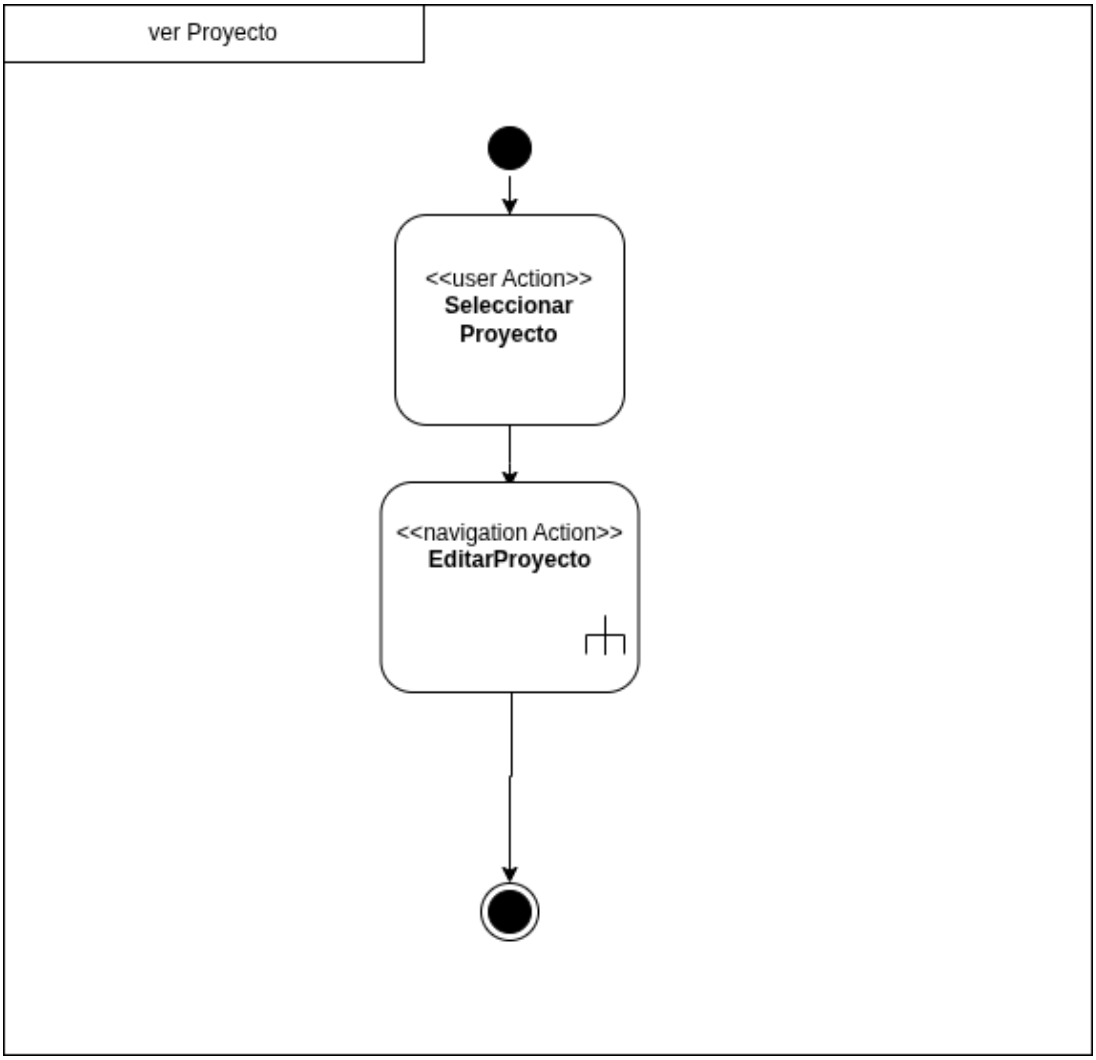

Figura 10: Diagrama de Actividad - Ver Proyecto

6.5.9. Diagrama de Actividad - Editar Proyecto

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Editar proyecto*13.

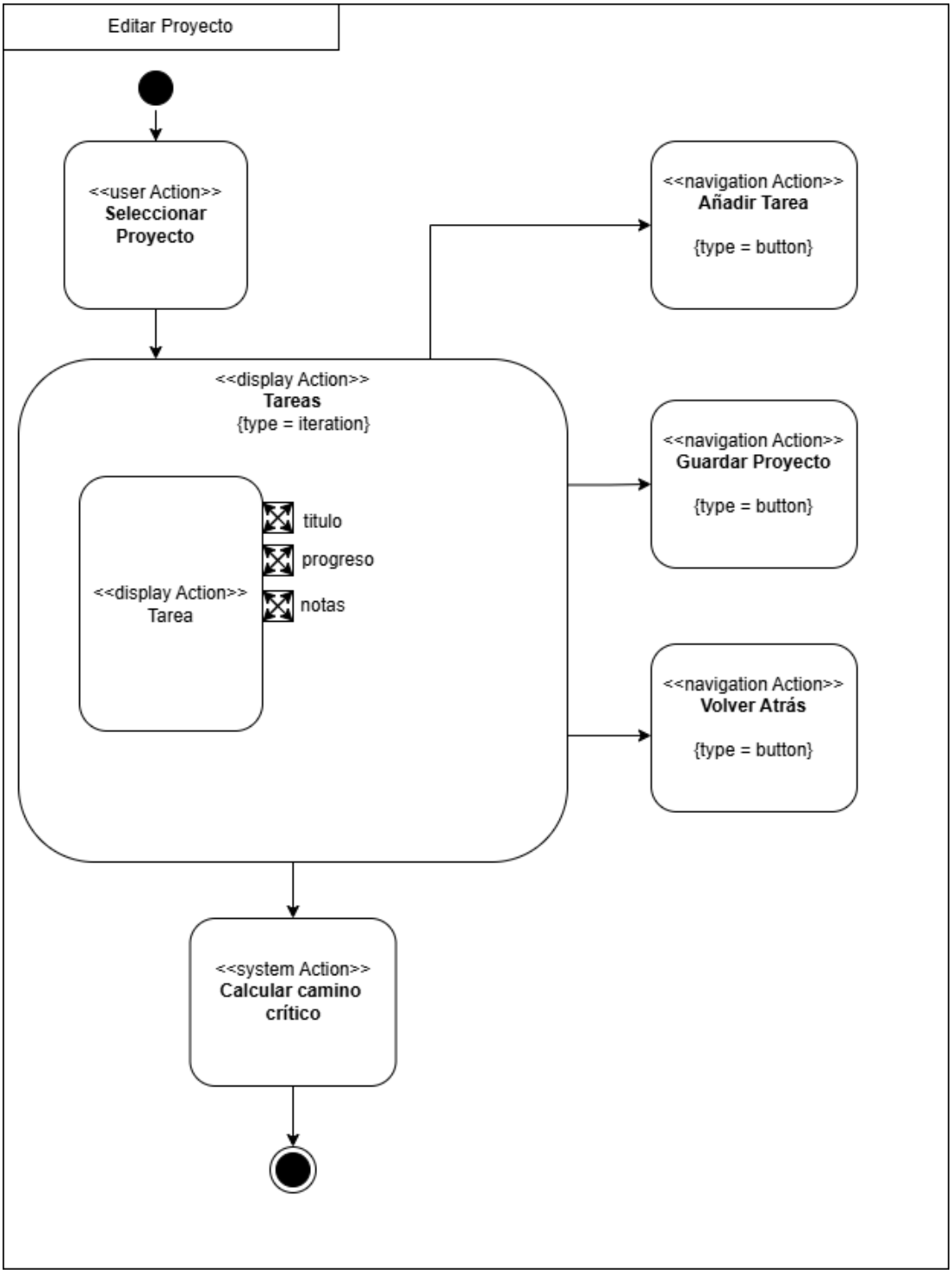

Figura 11: Diagrama de Actividad - Editar Proyecto

6.5.10. Diagrama de Actividad - Ver Resumen de un Proyecto

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Ver resumen de proyecto*14.

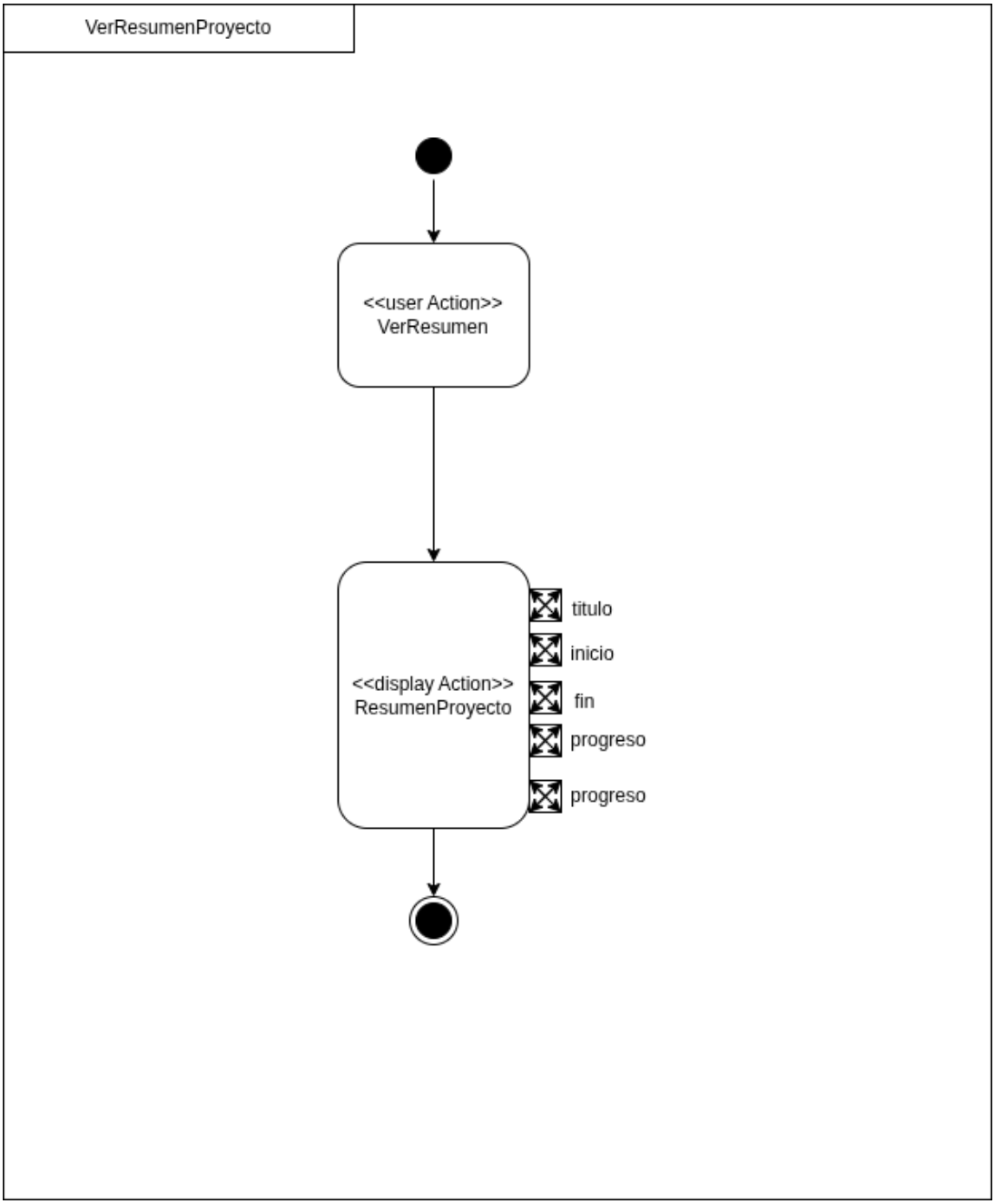

Figura 12: Diagrama de Actividad - Ver Resumen de un Proyecto

6.5.11. Diagrama de Actividad - Ver Proyectos

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Ver proyectos*15.

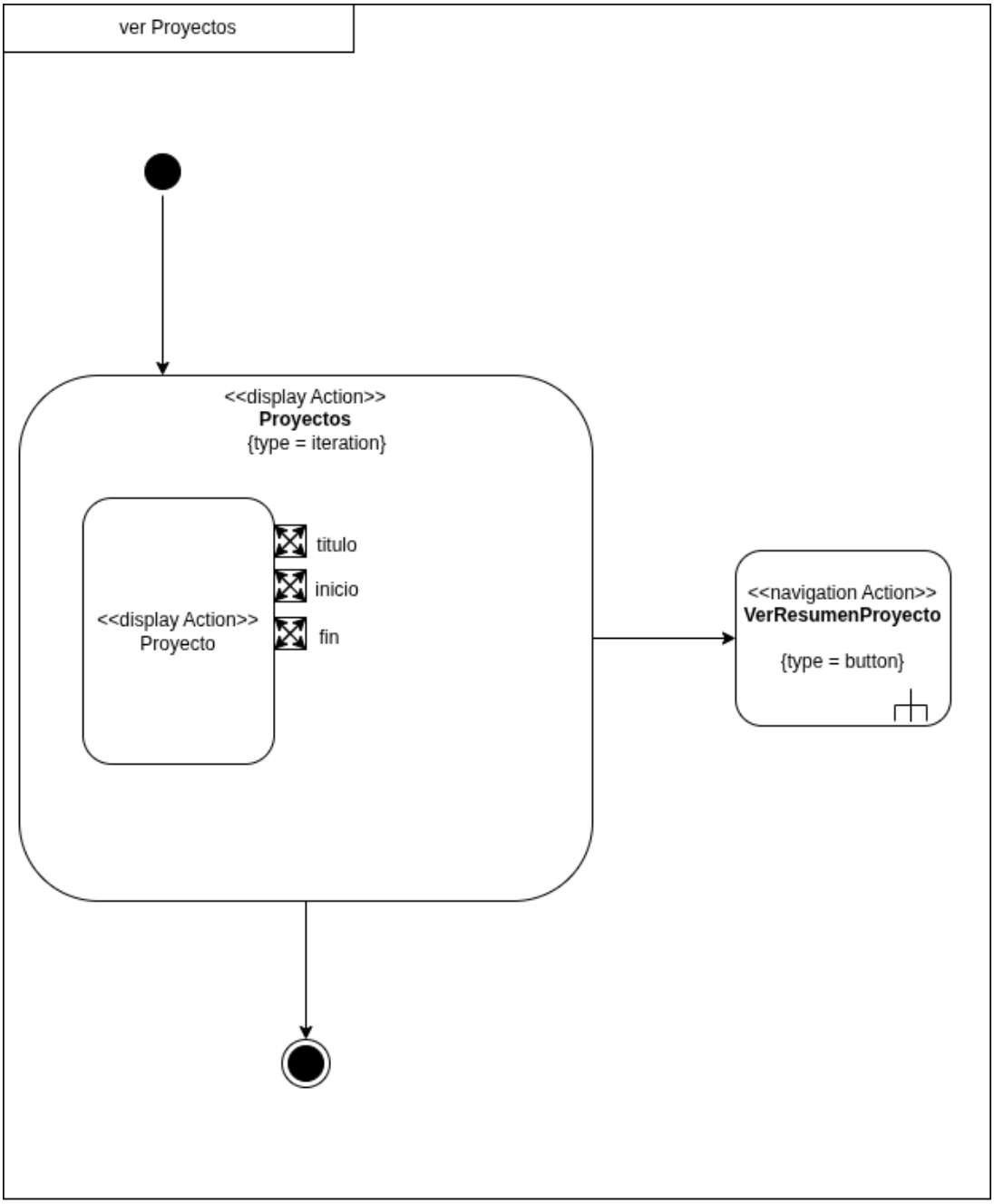

Figura 13: Diagrama de Actividad - Ver Proyectos

6.5.12. Diagrama de Actividad - Eliminar Proyecto

Este diagrama representa gráficamente el flujo de acciones del caso de uso *Eliminar proyecto*16.

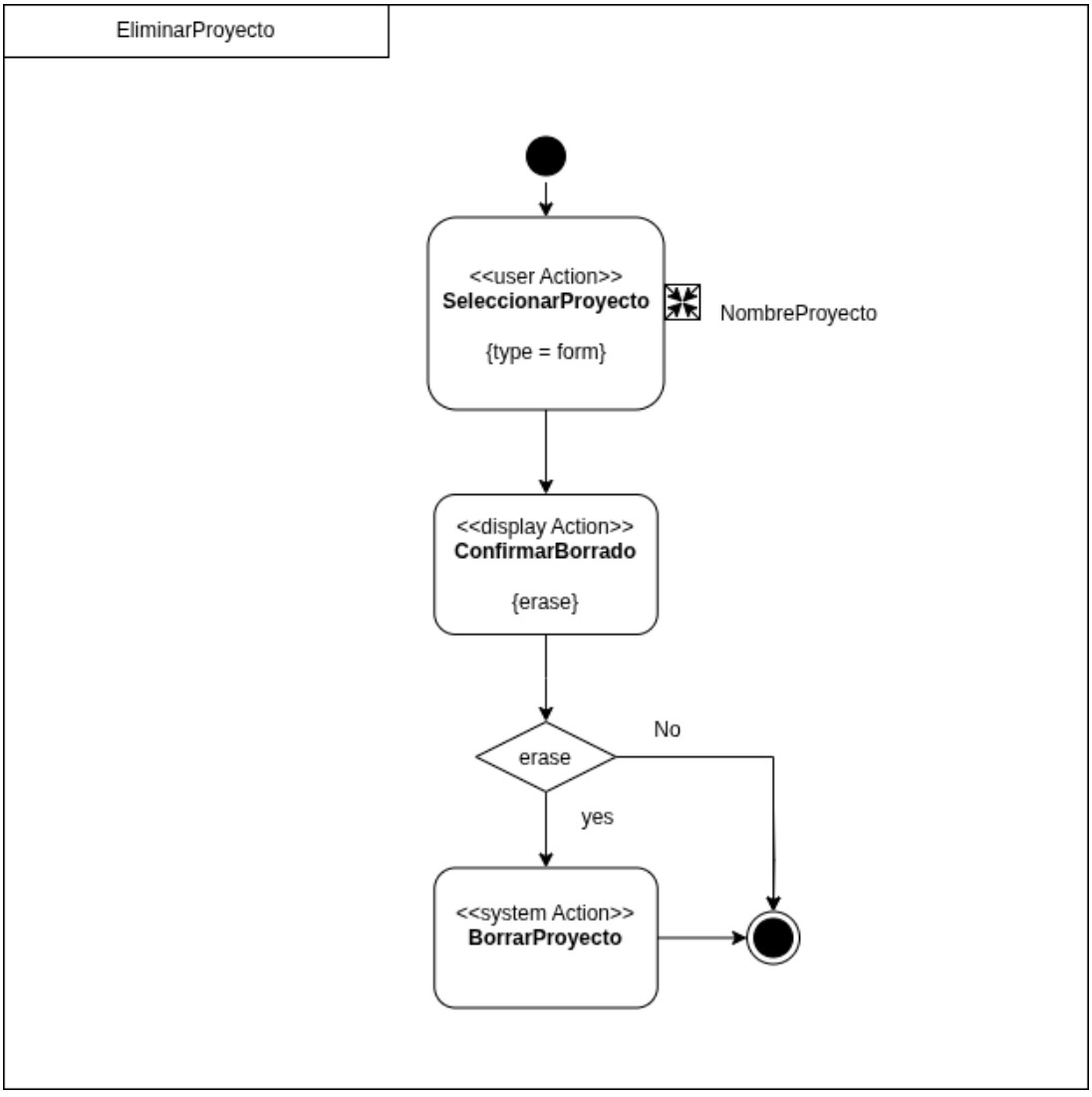

Figura 14: Diagrama de Actividad - Eliminar Proyecto

7. DISEÑO

En esta sección se presenta el proceso de diseño de la aplicación mediante la metodología UWE, partiendo de los requisitos funcionales y no funcionales definidos en el apartado de Análisis del Problema. El objetivo principal es describir la arquitectura global del sistema, así como las decisiones que han guiado la elaboración de cada uno de sus componentes.

7.1. Aspectos de la metodología UWE

Se utilizará la metodología UWE[5] *(UML-based Web Engineering)* como marco de referencia debido a su gran utilidad para el diseño de proyectos web al ser una metodología nacida precisamente para este entorno, dando forma a los próximos apartados.

Esta metodología fue diseñada para el desarrollo de aplicaciones web basadas en UML, permitiendo modelar y diseñar aplicaciones web mediante el uso de modelos y diagramas de UML, extendiendo mediante etiquetas, restricciones y estereotipos necesarios para describir conceptos propios de este tipo de sistemas, como la navegación, la presentación y la interacción[18]. su punto fuerte es que promueve un enfoque dirigido por modelos , lo que facilita estructurar el diseño desde una perspectiva clara y organizada, separando el contenido, la interfaz y los procesos de negocio, facilitando el diseño inicial, mantenimiento y la comprensión general del sistema a lo largo del tiempo.

Concretamente UWE define distintos modelos específicos que permiten representar las dimensiones de importancia de una aplicación web: el modelo de contenido[19] (para estructurar la información y los datos), el modelo de navegación[20] (para definir cómo se accede a dicha información), el modelo de presentación[21] (para representar la interfaz que verá el usuario) y el modelo de procesos[22] (para describir el comportamiento dinámico y las operaciones del sistema). Cada uno de estos modelos se relaciona con los requisitos identificados durante la fase de análisis[23], permitiendo trazar de forma directa cómo cada necesidad del sistema queda reflejada en el diseño.

7.2. Diagrama de contenido

El diagrama de contenido (especificado como un diagrama de clases) definirá las relaciones entre las diferentes entidades de datos, junto con sus atributos y relaciones entre dichas entidades que serán utilizadas en la aplicación.

El objetivo de este diagrama es establecer principalmente información que manejará la aplicación y la base de datos en su desempeño normal, además de la información que será servida al usuario en las diferentes vistas de la aplicación.

Así garantizamos un contenido con una organización y estructuración coherente y respondiendo a las necesidades funcionales del dominio del problema, además sienta las bases de una navegación eficiente entre las vistas de la aplicación, ya que nos permite ver la información que interesa manejar a cada vista.

En base a la funcionalidad establecida en los diagramas de actividad así como a los requisitos de información y no funcionales, se almacenará la información necesaria como las estructuras definidas en la figura 15.

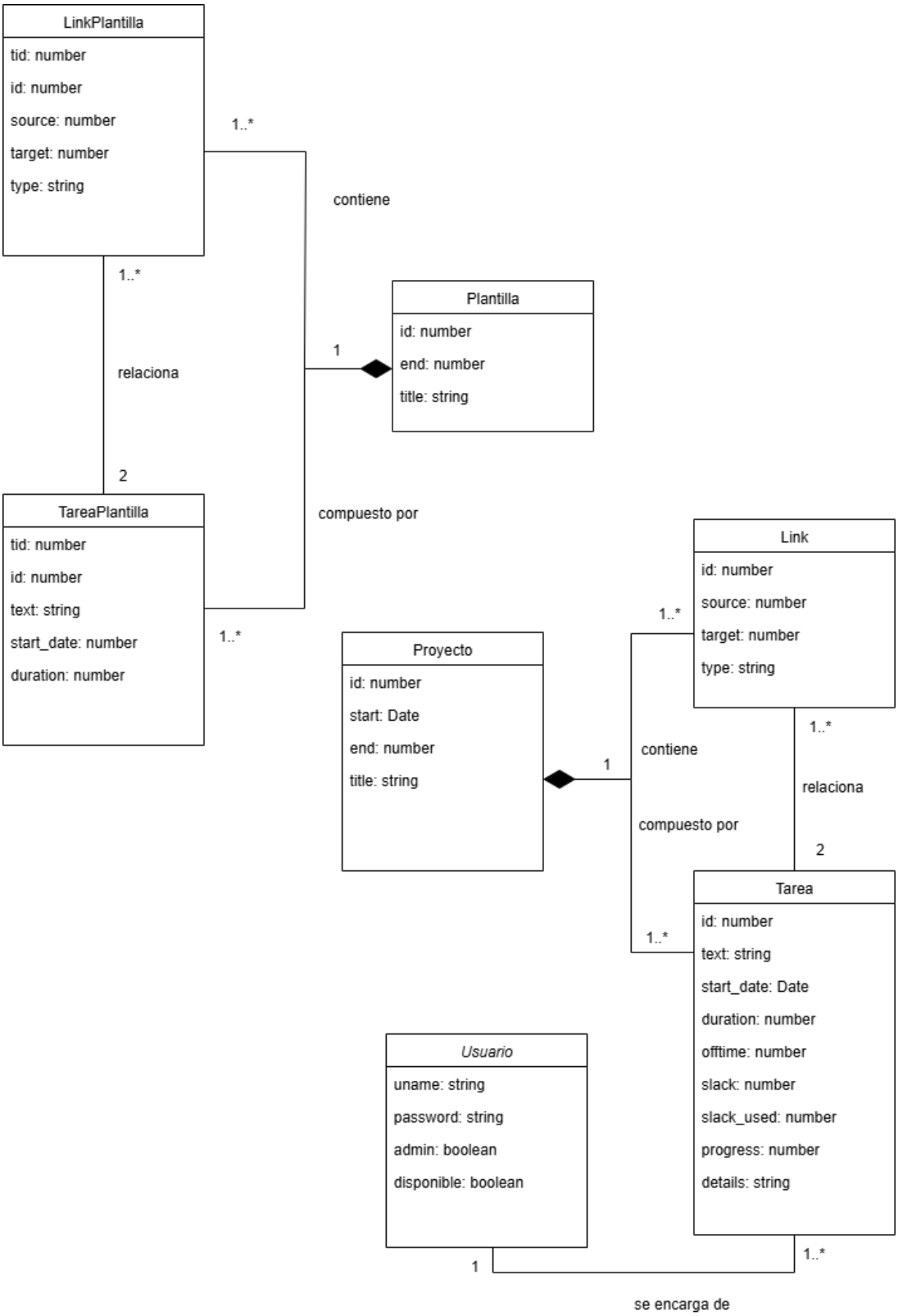

Figura 15: Diagrama de clases de Contenido

7.2.1. Clase Usuario

clase que representa los atributos y responsabilidades principales de los usuarios de la aplicación. Cada usuario se puede encargar de varias tareas.

Sus atributos son:

- **uname**: Nombre del usuario dentro de la aplicación, con restricciones sobre su formato (solo minúsculas).
- **password**: Contraseña del usuario dentro de la aplicación, se le aplican restricciones sobre su formato (debe contener minúsculas, mayúsculas y números).
- **admin**: Determina si el usuario puede iniciar sesión en la aplicación
- **disponible**: Indica si un usuario puede asignarse a tareas.

7.2.2. Clase Tarea

Representa los atributos de las tareas, asignadas a usuarios y relacionadas con otras tareas mediante un enlace, cada tarea pertenece a un proyecto.

Sus atributos principales son:

- **id**: Identificador de la tarea.
- **text**: Nombre de la tarea.
- **start_date**: Fecha de inicio de la tarea.
- **duración**: Duración de la tarea en minutos.
- **offtime**: Tiempo no laboral que ocupará la tarea.
- **slack**: Holgura de la tarea.
- **slack_used**: Cantidad de holgura usada hasta la fecha.
- **progress**: Indica el progreso de la tarea.
- **details**: Observaciones durante la realización de la tarea.

7.2.3. Clase Link

Representa la relación entre una tarea y otra.

Sus atributos son:

- **id**: Identificador del enlace.
- **source**: Identificador de la tarea origen del enlace.
- **target**: Identificador de la tarea destino del enlace.
- **type**: Tipo de enlace, por defecto de fin del origen a inicio del destino.

7.2.4. Clase Proyecto

Representa un proyecto en la aplicación, conformado por varias tareas y enlaces entre ellas.

Sus atributos principales son:

- **id**: Identificador del proyecto.
- **start**: Fecha de inicio de la tarea.
- **end**: Horas desde el inicio que dura el proyecto.
- **titulo**: Nombre del proyecto.

7.2.5. Clase TareaPlantilla

Representa los atributos de las tareas, relacionadas con otras tareas mediante un enlace, cada tarea pertenece a una plantilla de proyectos.

Sus atributos principales son:

- **id**: Identificador de la tarea.
- **text**: Nombre de la tarea.
- **start_date**: tiempo en horas desde el inicio de la plantilla.

7.2.6. Clase LinkPlantilla

Representa la relación entre una tarea y otra de una misma plantilla de proyectos.

Sus atributos son:

- **id**: Identificador del enlace.
- **source**: Identificador de la tarea origen del enlace.
- **target**: Identificador de la tarea destino del enlace.
- **type**: Tipo de enlace, por defecto de fin del origen a inicio del destino.

7.2.7. Clase Plantilla

Representa un proyecto en la aplicación, conformado por varias tareas y enlaces entre ellas.

Sus atributos principales son:

- **id**: Identificador de la plantilla de proyectos.
- **end**: duración base de la plantilla de proyectos.
- **titulo**: Nombre de la plantilla de proyectos.

7.3. Diagrama de navegación

El objetivo del diagrama de navegación es planear como se deberá mover el usuario por las diferentes vistas de la aplicación.

Para dar un visión clara de estos movimientos entre las vistas de la aplicación, el diagrama establece a estas como una serie de nodos

Cada nodo dentro del diagrama representa una vista concreta que el usuario puede alcanzar durante su interacción con la aplicación. Estas vistas pueden ser pantallas de inicio, formularios, listados, entre otros, y su disposición dentro del diagrama permite identificar el flujo principal de navegación, así como rutas alternativas.

Estos nodos quedan comunicados entre si por enlaces, de navegación o de proceso. Los primeros describen que se puede realizar una comunicación a partir de acciones simples en una vista, mientras que los segundos establecen que la comunicación solo se podrá realizar tras algún tipo de procesamiento. Con todo esto explicado, el diagrama de navegación queda reflejado en la figura16.

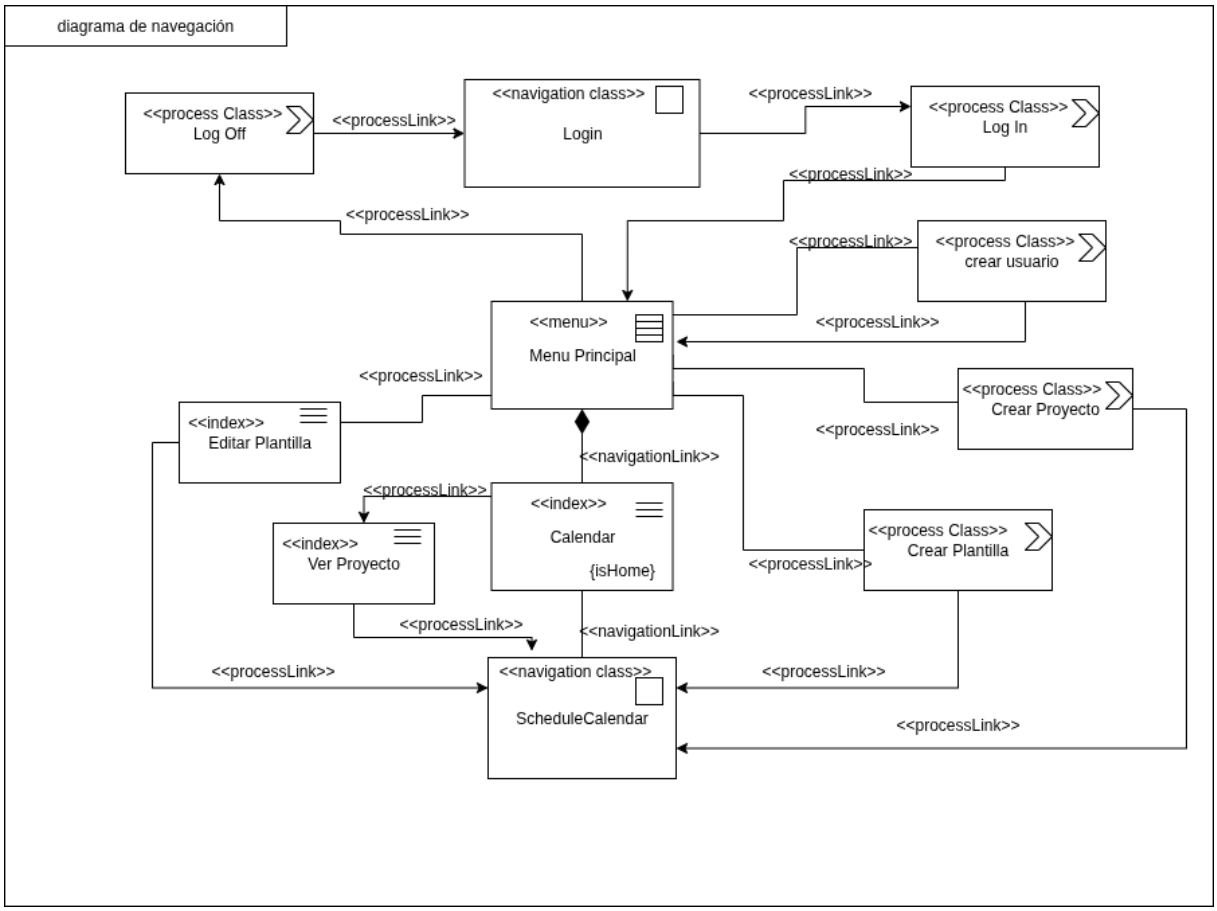

Figura 16: Diagrama de Navegación

En este diagrama podemos ver las posibilidades de navegación principales y mas importantes disponibles a los usuarios. Se puede ver como por ejemplo la vista de calendario (que contiene un índice de proyectos) se visualiza al mismo tiempo que el menú principal, dependiendo su existencia de la del menú principal.

Además se controla la navegación entre las diferentes vistas. No se permite un tráfico libre entre ellas, haciendo que diferentes procesos verifiquen condiciones que se deben cumplir para navegar entre ellas.

Tomemos el ejemplo de navegación entre la vista Login y de Menú Principal, para ir de la primera a la segunda se debe ejecutar el proceso *Log In*, encargado de verificar si se tienen credenciales válidas para acceder a la aplicación. Del mismo modo, el proceso *Log Off* controla que solo se pueda volver a la vista de Login cuando ya no hay una sesión activa de un usuario.

en el caso de la navegación entre Menú Principal/Calendar hacia ScheduleCalendar, podemos llegar a esta ultima a traves de tres casos concretos:

1. Al Crear una Plantilla mediante el proceso Crear Plantilla.
2. Al Editar una plantilla elegida en la vista índice.
3. Al Crear un proyecto (proceso Crear Proyecto) mediante una plantilla.
4. Al visualizar un proyecto del índice de Calendar.

7.4. Diagramas de presentación

Con los siguientes diagramas de presentación se tiene el objetivo de establecer como estarán organizados los elementos gráficos dentro de cada vista de la aplicación, En otras palabras nos da un prototipado de la interfaz de usuario, proporcionando elementos que guiarán en el diseño, tipando el tipo de interacción que se espera que un usuario realice en una vista

Tomando de entrada la información de los diagramas de contenido y navegación, los siguientes diagramas de presentación proporcionarán una interpretación lógica y coherente de como deberán ser representadas las estructuras de datos de la aplicación al usuario.

7.4.1. Diagrama de presentación - Login

Figura 17: Diagrama de Presentación - Login

Esta vista presentará al usuario los campos necesarios para que un usuario introduzca sus credenciales de acceso y las envíe al servidor para su verificación, ademas de un mensaje de error en caso de que las credenciales no sean válidas.

7.4.2. Diagrama de presentación - Calendario

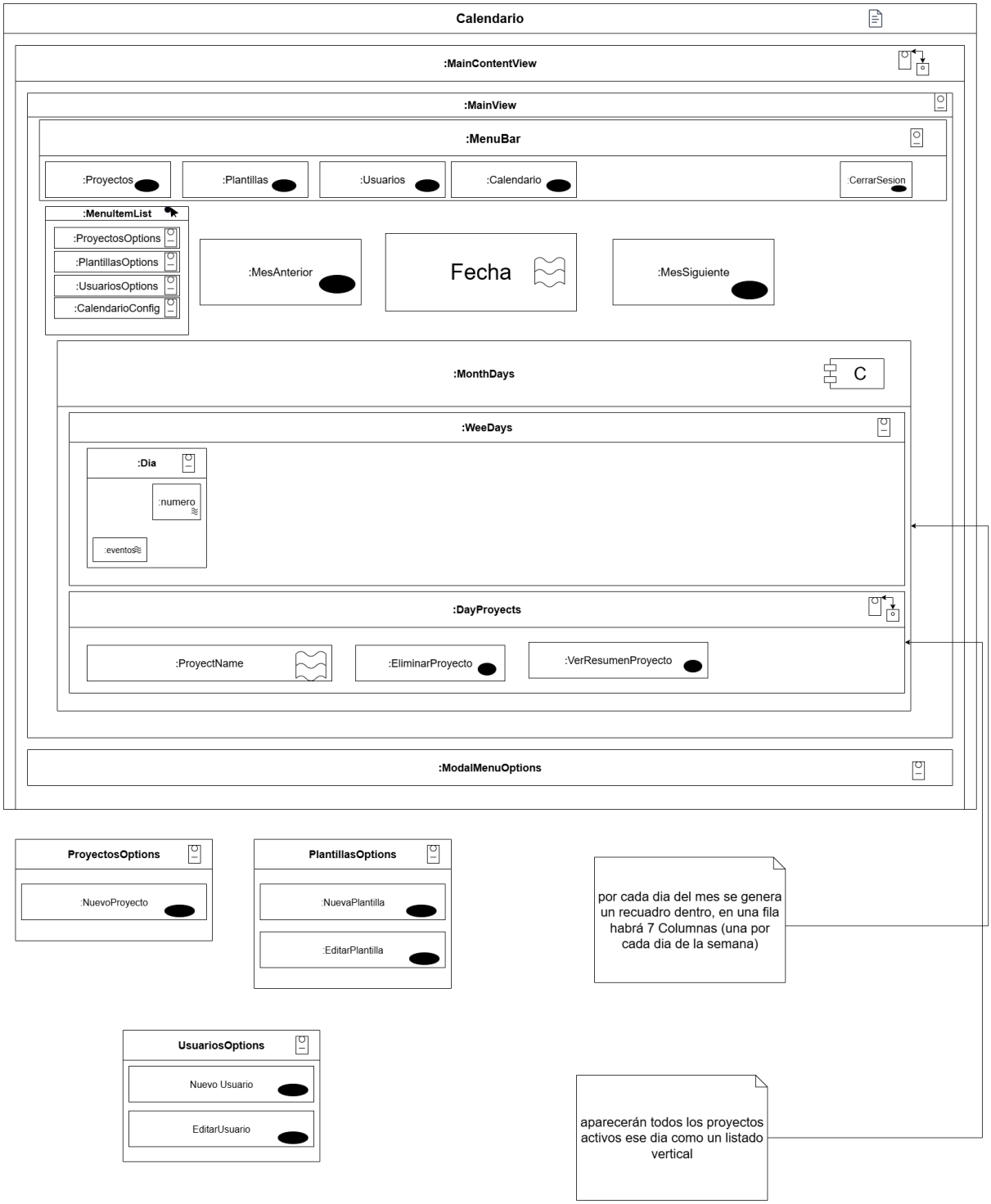

Figura 18: Diagrama de Presentación - Calendario

Esta vista muestra una barra de menú principal ademas de la vista principal del calendario. Empezando con el calendario, se muestran los proyectos de este mes,

representando los días y los proyectos que ocupan su tiempo. Los botones laterales cambian el mes que se representa. Si elegimos un día, se amplía y muestra los proyectos y las opciones que se pueden realizar sobre ellos.

Con respecto a la barra de menú, muestra opciones principalmente con respecto a la creación de plantillas de proyectos y de proyectos en si, al hacer click en una opción se muestra un menú desplegable, cada uno con sus propias opciones para mostrar nuevas vistas.

7.4.3. Diagrama de presentación - Ventana modal

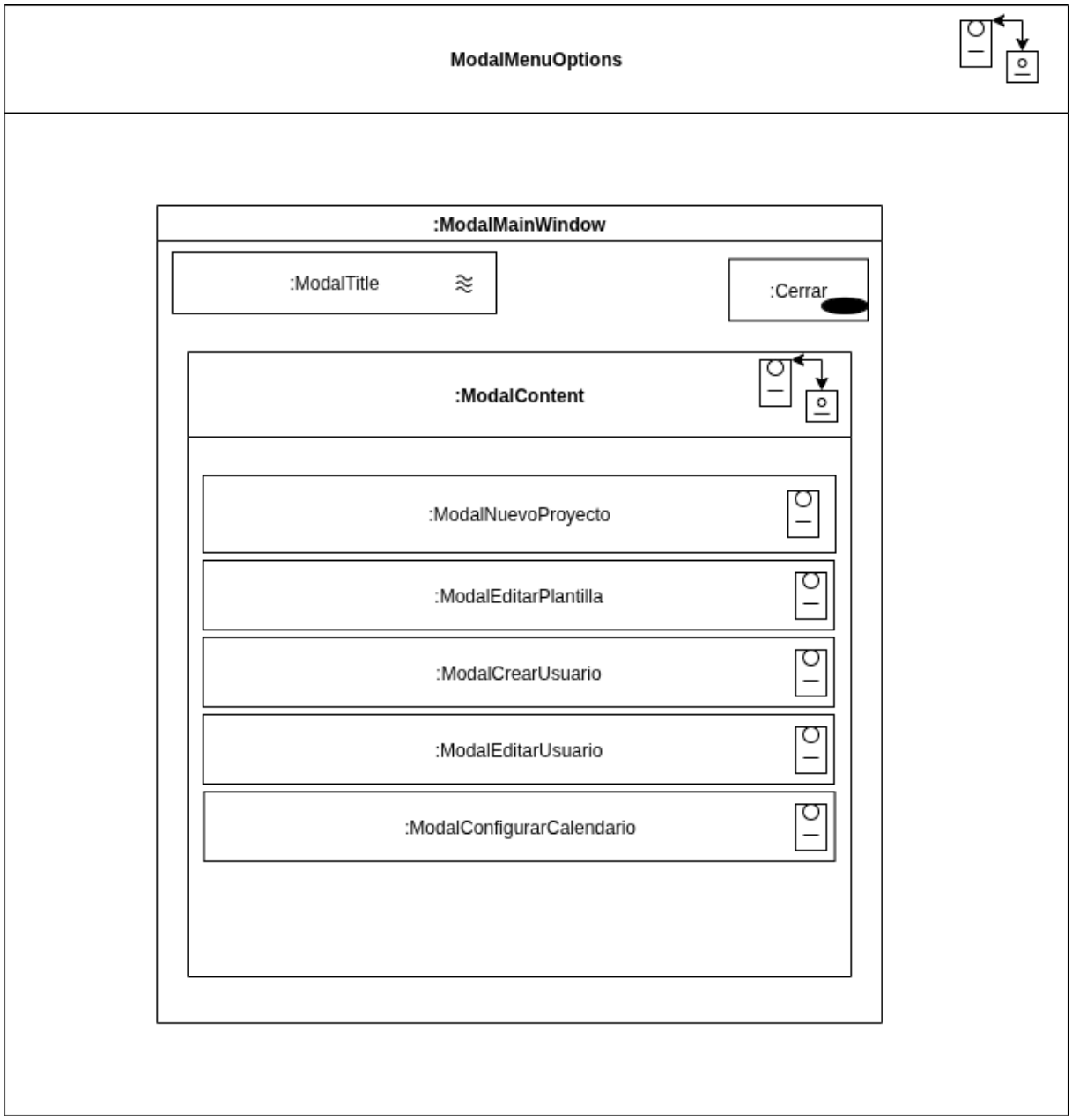

Figura 19: Diagrama de Presentación - Ventana modal

Es la ventana flotante que se muestra cuando se selecciona una opción de uno de los menús desplegables de la barra de menú. todas las opciones del menú comparten ventana, alternando solamente el contenido mostrado en esta en función de la la opción elegida, el contenido de esta ventana.

7.4.4. Diagrama de presentación - Opciones de ventana modal

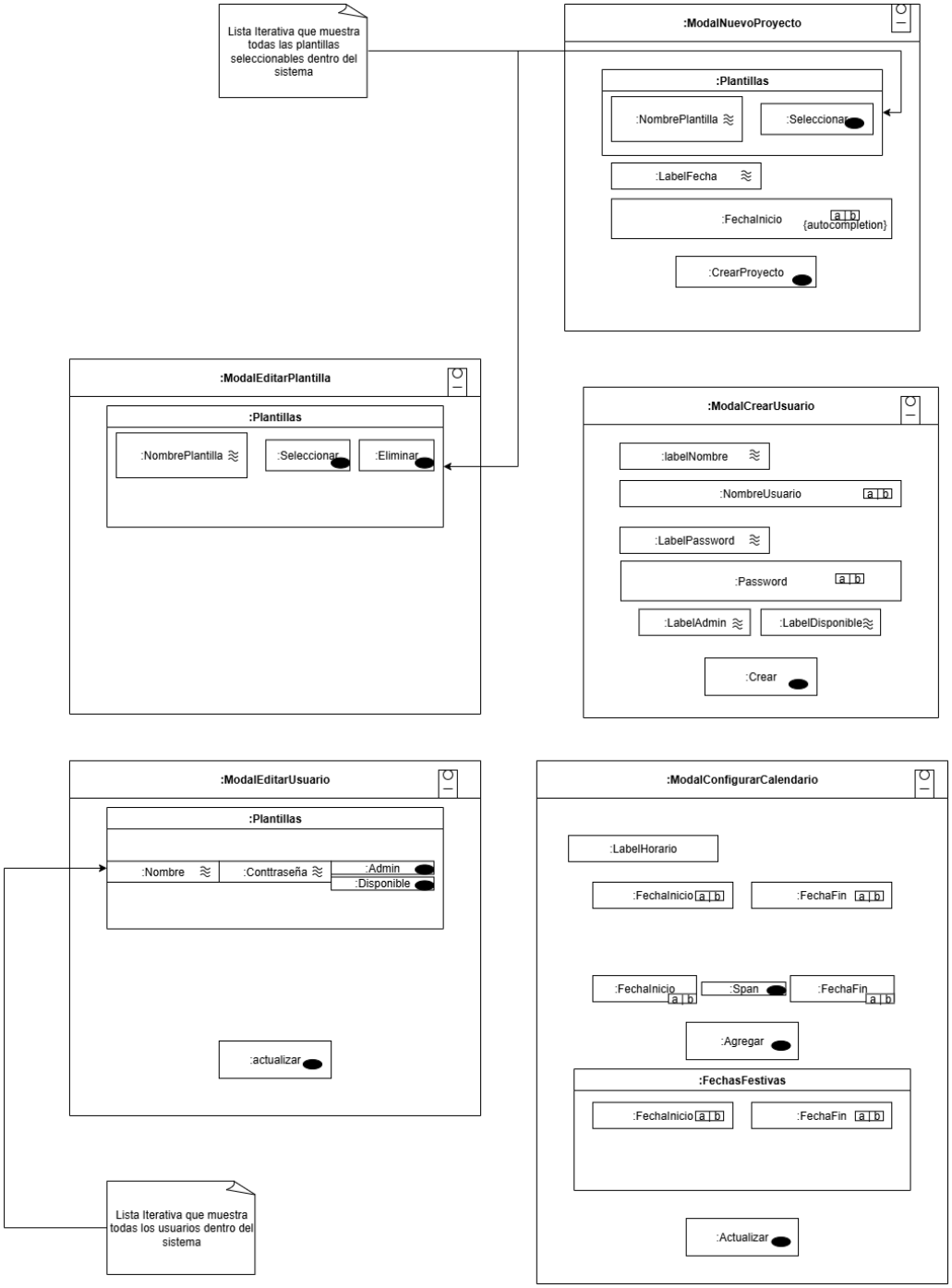

Figura 20: Diagrama de Presentación - Opciones de ventana modal

Representa el conjunto de elementos que se mostrarán en la ventana flotante en

función de la opción elegida, se tiene así, para una misma ventana, formularios para Editar una plantilla, Editar un proyecto, Crear una Plantilla o proyecto, añadir y editar usuarios..., etc.

por ejemplo, para crear un nuevo proyecto, la ventana muestra información de las plantillas de proyectos de la aplicación para elegir una, y un campo para elegir la fecha de inicio del proyecto

También para modificar un proyecto, la ventana mostrará un listado de todas las plantillas de la aplicación y las operaciones que se pueden realizar sobre ellas.

7.4.5. Diagrama de presentación - Schedule Calendar

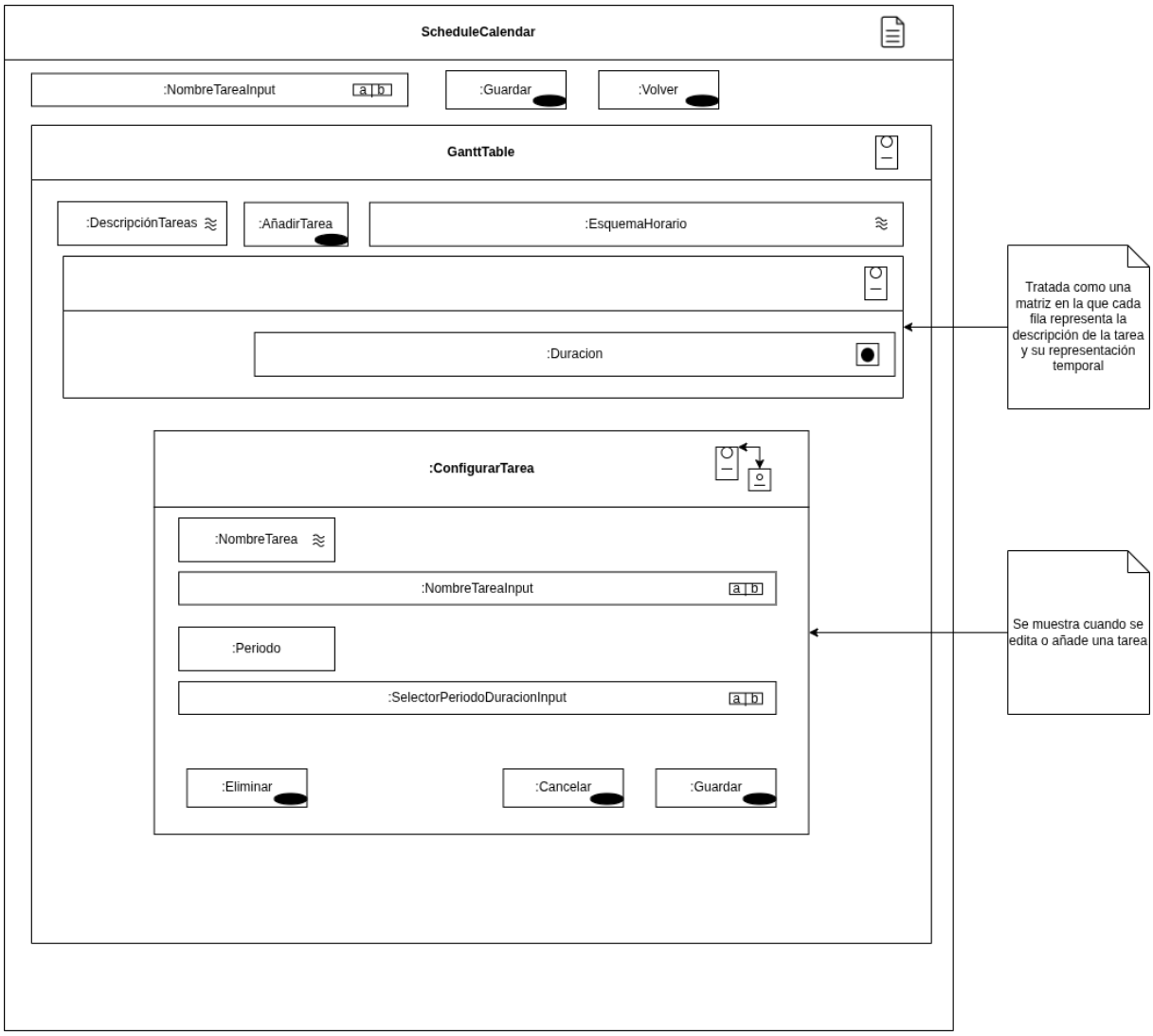

Figura 21: Diagrama de Presentación - Schedule Calendar

La representación de esta vista es mas sencilla de lo que parece, en esta muestro un conjunto de utilidades en la parte superior para volver a la vista anterior, guardar el proyecto o plantilla de proyecto, y asignarle un nombre. Debajo se representa un

diagrama de Gantt mediante barras que representa cada tarea y el periodo de tiempo que ocupa. Haciendo click en una tarea se muestra su ventana de configuración, que nos muestra su fecha de inicio, duración, nombre, operarios encargados de ella y observaciones sobre la tarea.

7.5. Diagramas de procesos

Los diagramas de procesos nos van a permitir describir las acciones internas que ocurren durante la ejecución de una funcionalidad (caso de uso).

Gracias a estos diagramas se pueden especificar los pasos a seguir para desarrollar una funcionalidad sin entrar en detalles de implementación, pero siendo coherente con la representación del contenido y el el flujo de acciones, ademas de añadir un nivel de detalle mayor al tratarse ahora de una representación de las acciones transparentes al usuario que realiza el sistema, lo que nos permite comprobar de primera mano la complejidad de las acciones planteadas en un inicio y agruparlas según los objetos de datos que requerirán usar y las señales que recibirán. Es decir, que si los diagramas de actividad representaban el flujo de acciones que realiza el usuario interactuando con el sistema, los diagramas de procesos mostrarán como trabaja el sistema de para proporcionar una funcionalidad transparente al usuario

7.5.1. Diagrama de Procesos Global

En el diagrama de la figura22 se muestran todos los procesos de interés para el desarrollo de la aplicación, organizados en una jerarquía.

No hay que confundir esta jerarquía puramente organizativa con el funcionamiento real del sistema, ya que lo que hace es agrupar procesos según la información principal que manipulan (operaciones sobre usuarios, plantillas o proyectos). Además, el hecho de que todas hereden directa o indirectamente del proceso de notificaciones indica que de cara a la experiencia de usuario, el resultado de estas operaciones será mostrado al usuario.

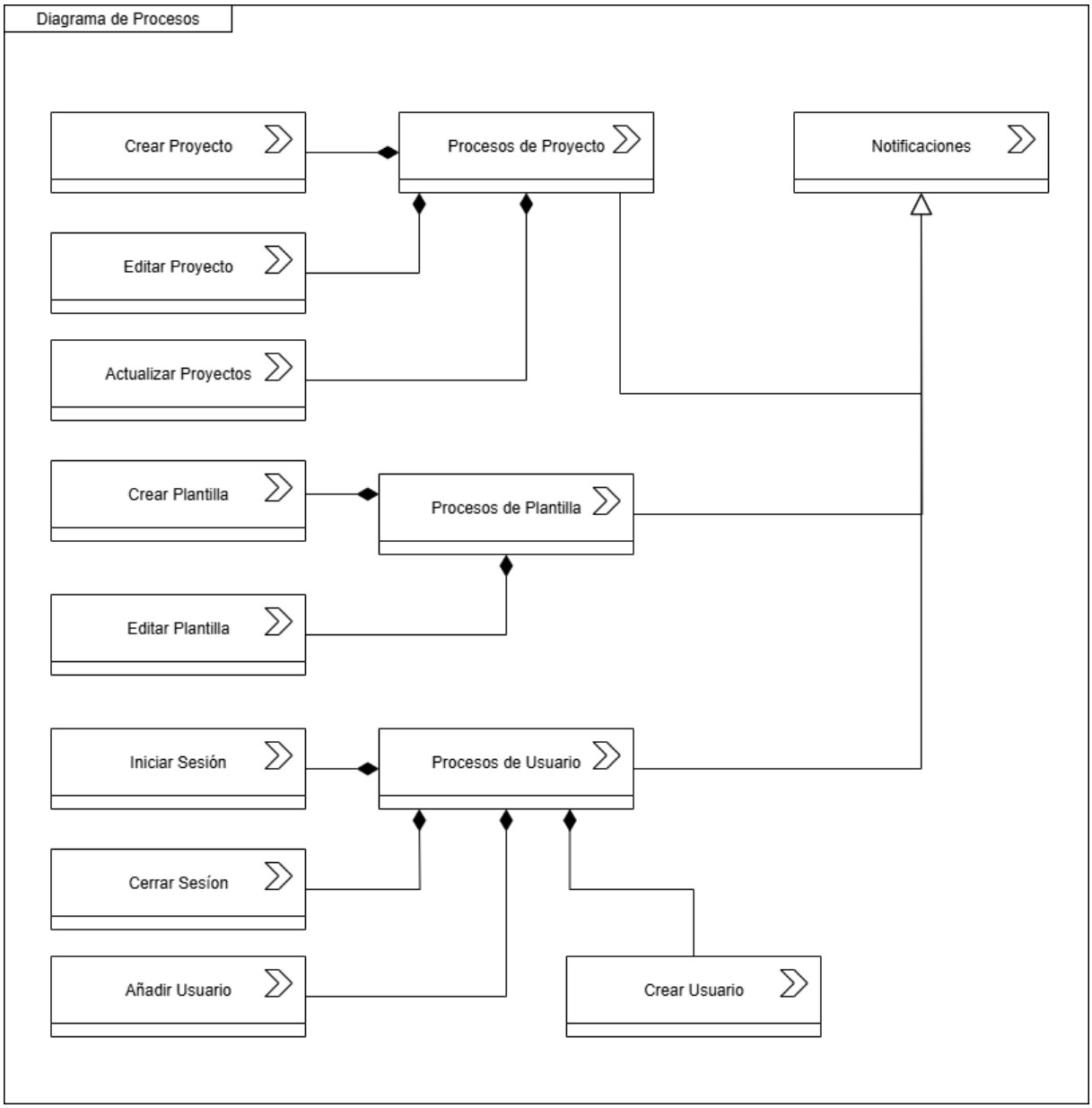

Figura 22: Diagrama de Procesos Global

7.5.2. Diagrama de Proceso - Iniciar Sesión

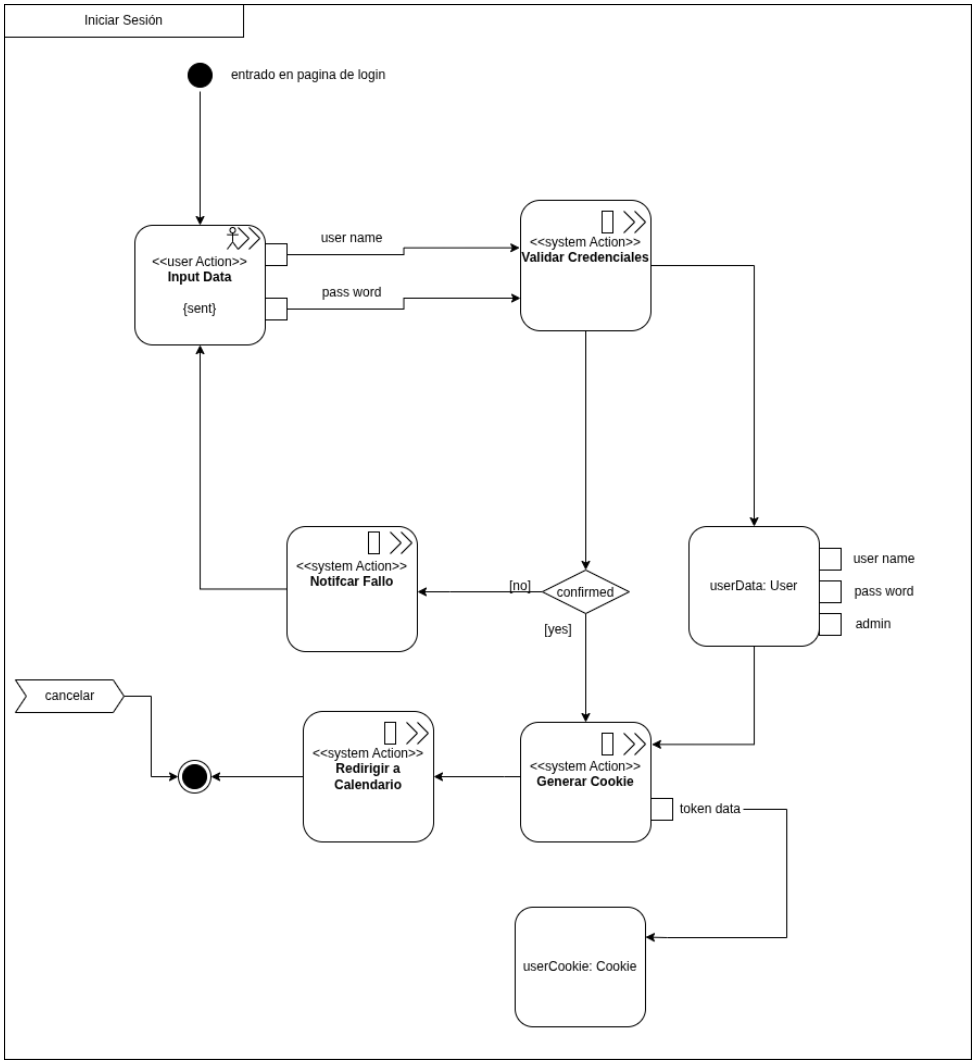

Figura 23: Diagrama de Proceso - Iniciar Sesión

Tal como se muestra en la figura23, el usuario introduce sus datos, el sistema los valida, y con esos datos válidos, se genera una *cookie* que representará la capacidad para permanecer con una sesión en la aplicación, en caso de no ser datos validos, el sistema pide volver a introducir los datos.

7.5.3. Diagrama de Proceso - Cerrar Sesión

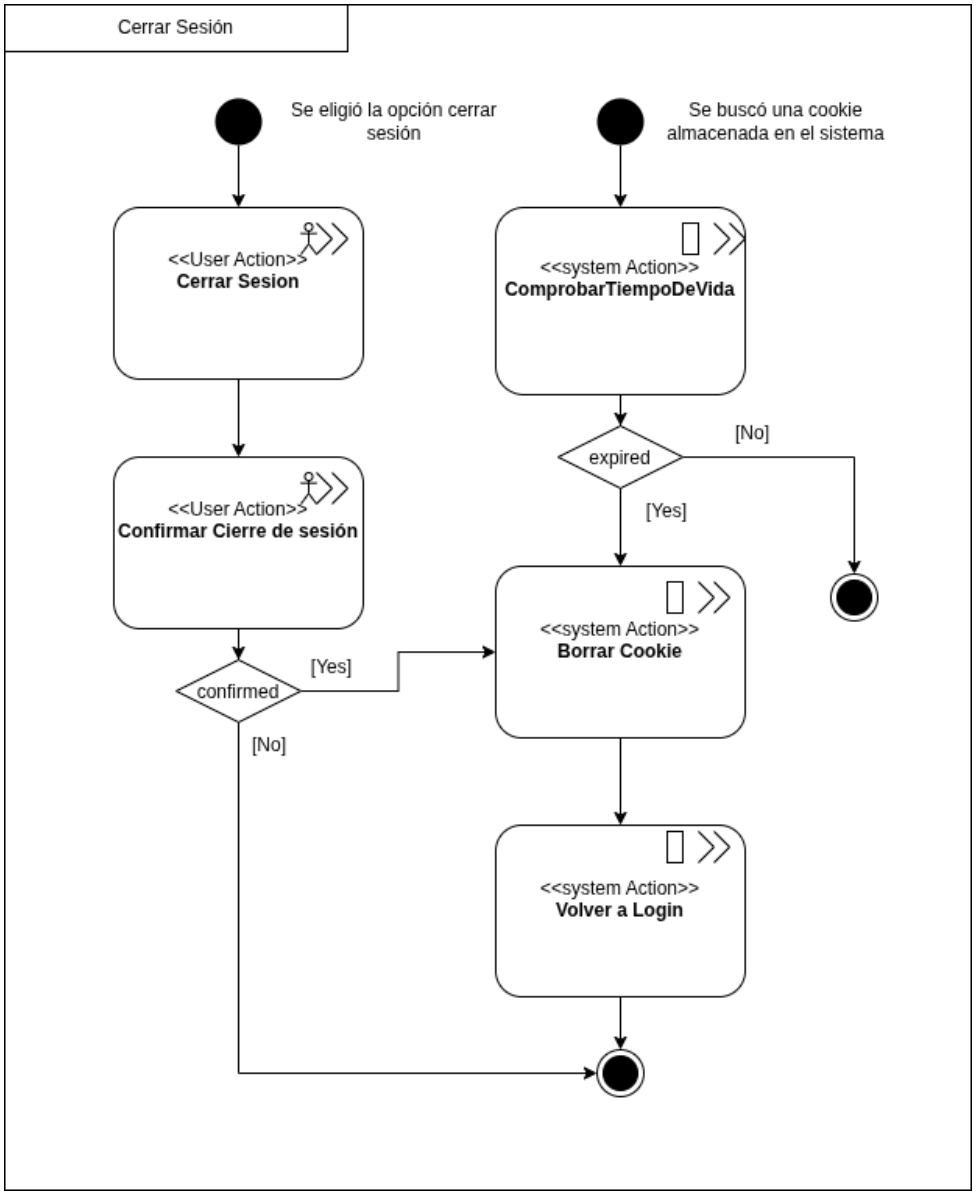

Figura 24: Diagrama de Proceso - Cerrar Sesión

En La figura24 representa las formas en las que se puede cerrar la sesión de usuario. El usuario puede bien cerrar la sesión manualmente, o puede ser que expire la cookie que mantiene la sesión abierta. en ambos casos se elimina y se vuelve a la pantalla inicial.

7.5.4. Diagrama de Proceso - Crear Usuario

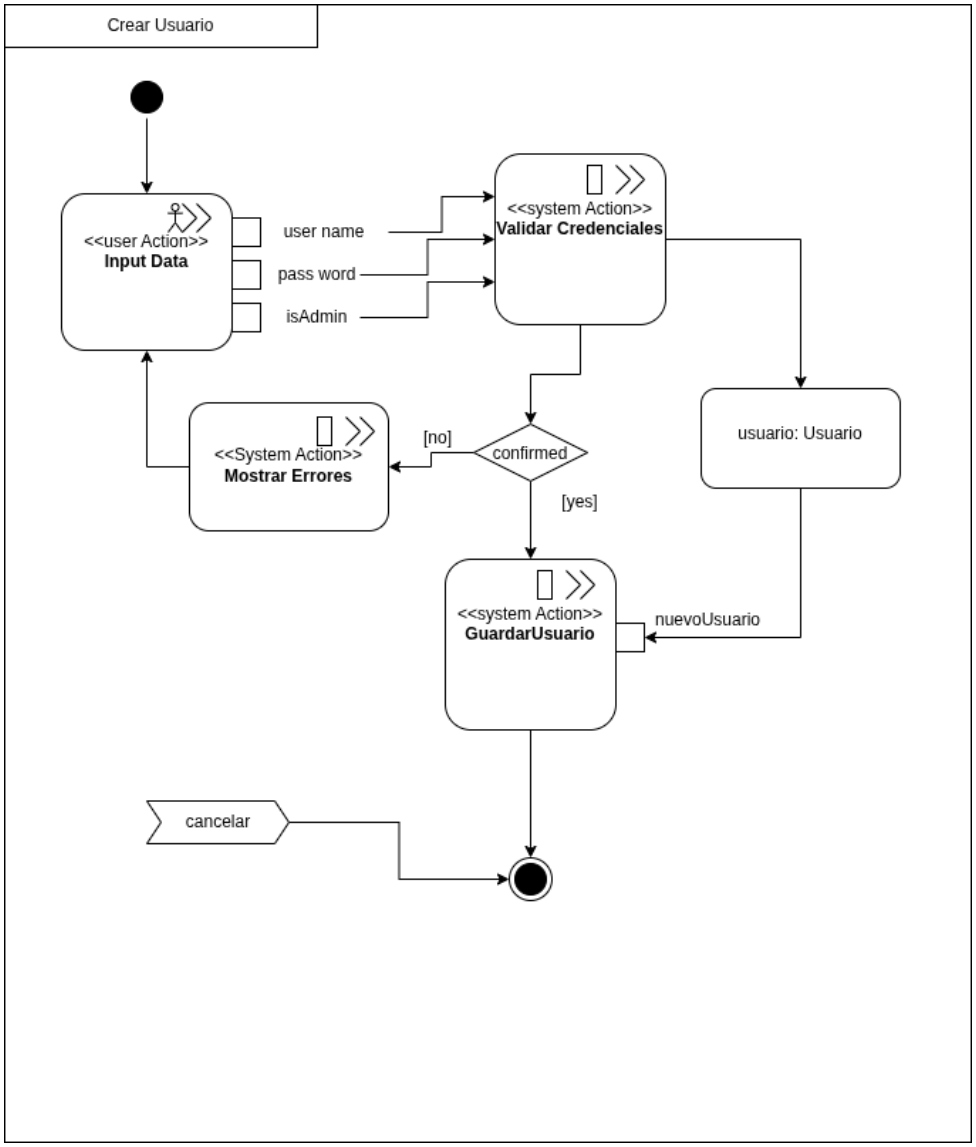

Figura 25: Diagrama de Proceso - Crear Usuario

En la figura25 se trata la creación un nuevo usuario. Un administrador introduce credenciales, el sistema las comprueba y si cumplen los requisitos, el usuario es creado y guardado, en otro caso se muestran los errores en las credenciales del usuario y se pide al administrador volver a introducirlas.

7.5.5. Diagrama de Proceso - Añadir Usuario

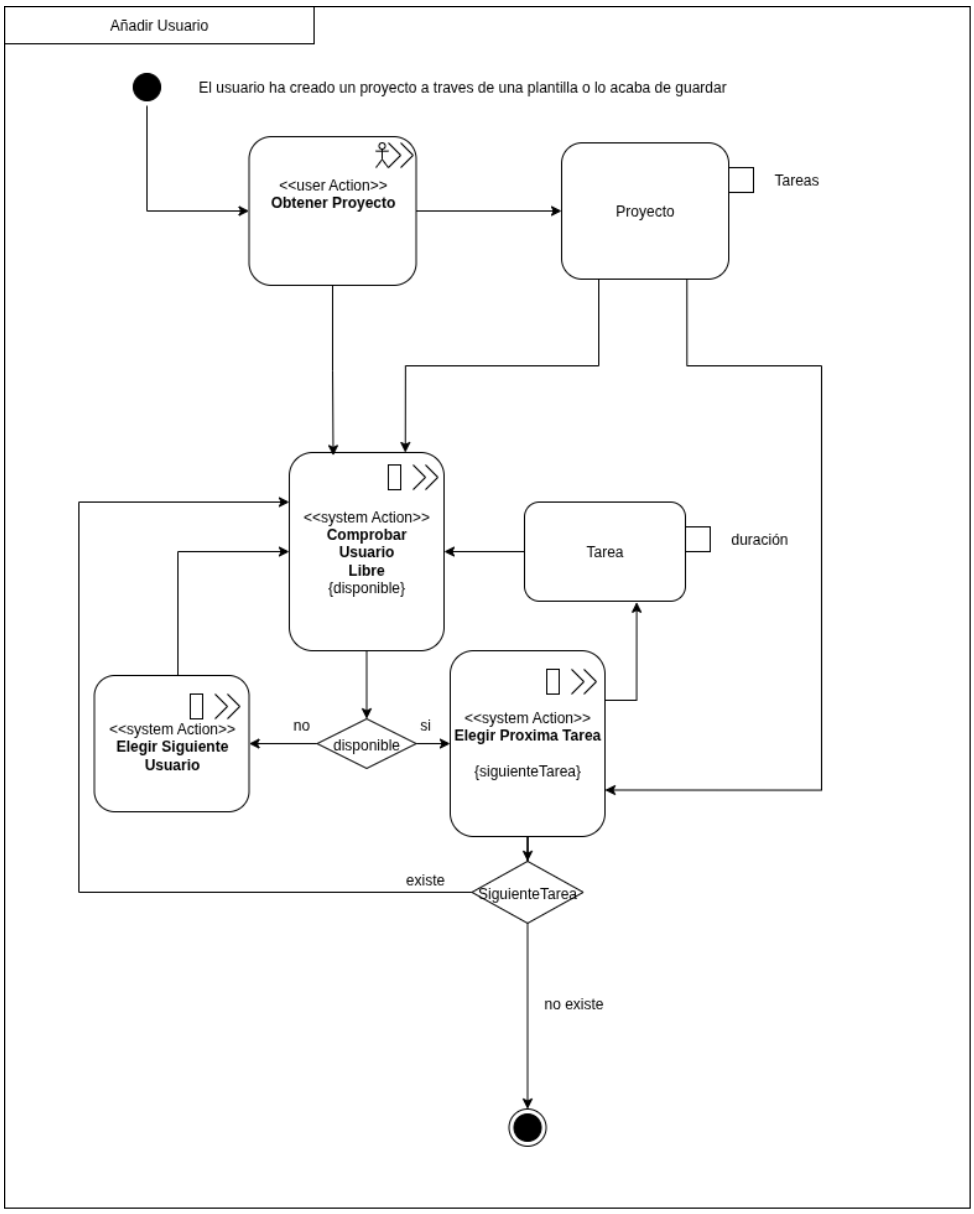

Figura 26: Diagrama de Proceso - Añadir Usuario

En la figura26 se muestra como se añaden los usuarios necesarios a las tareas del proyecto en creación. Una vez obtenido el proyecto (debido a que un administrador lo esta creando), se obtienen las tareas del mismo, iterativamente se va eligiendo una tarea a la que se debe asignar uno o mas usuarios, se comprueba usuario a usuario que esté libre para ejecutar la tarea, cuando ya se han asignado todos los usuarios necesarios a una tarea, se pasa a la siguiente, cuando acabamos todas las tareas, acaba el proceso, si no hay usuarios que se puedan encargar de una tarea, el proceso avisa al usuario y finaliza sin crear el proyecto.

7.5.6. Diagrama de Proceso - Crear Plantilla

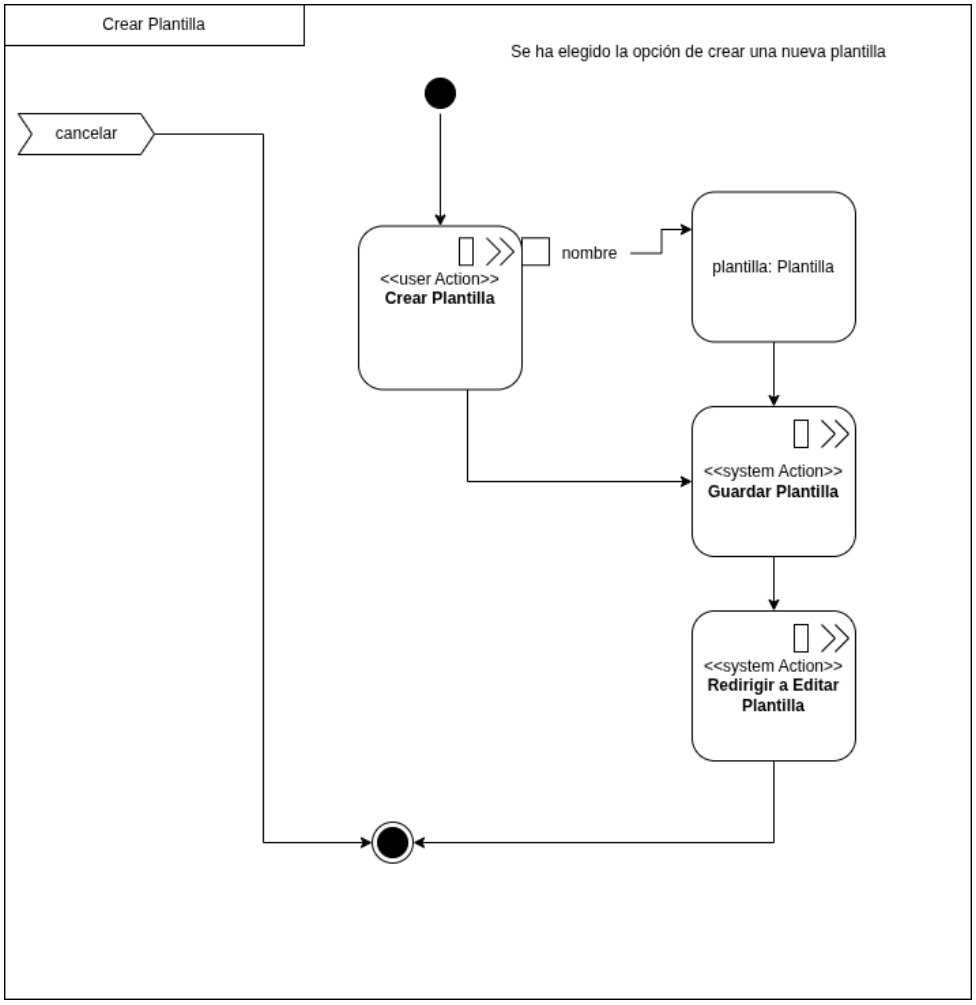

Figura 27: Diagrama de Proceso - CrearPlantilla

La figura27 muestra el proceso de creación de una plantilla. Cuando un administrador selecciona esa opción, se crea la plantilla, se guarda, y luego se muestra su proceso de edición.

7.5.7. Diagrama de Proceso - Editar Plantilla

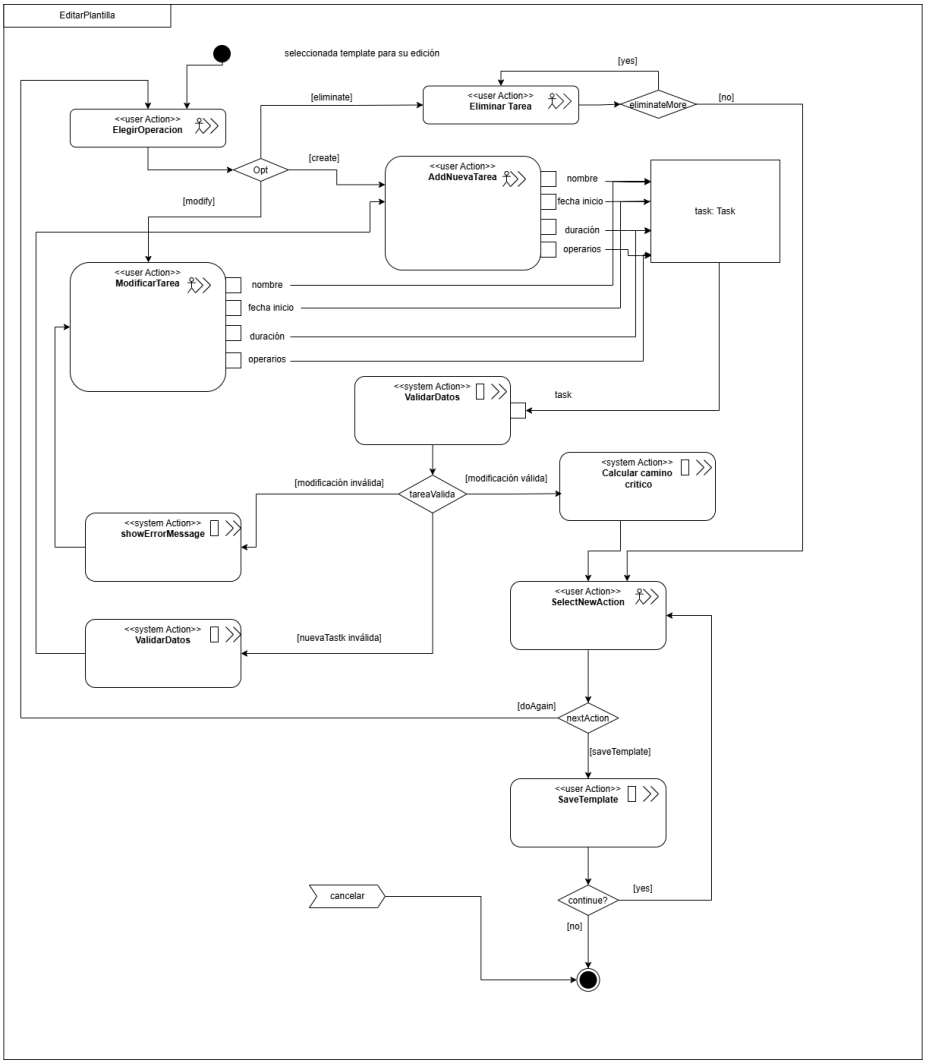

Figura 28: Diagrama de Proceso - Editar Plantilla

Esta figura28 nos muestra las actividades que se pueden desencadenar en función de lo que quiera hacer el administrador. En concreto puede:

1. Añadir una nueva tarea al proyecto indicando sus datos, si estos son válidos, permite al usuario realizar una nueva acción.

2. Modificar una tarea existente, para ello edita los datos de una tarea que ya existiese en el proyecto, si la modificación es valida, permite al administrador realizar una nueva acción.

3. Eliminar una tarea existente, elige una tarea y confirma su intención de borrarla, tras lo cual el sistema la borra, y permite al administrador elegir hacer otra acción.

En cualquiera de los casos, al elegir una nueva acción a realizar, puede elegir guardar las modificaciones, o cancelar el proceso en cualquier punto.

7.5.8. Diagrama de proceso - Crear Proyecto

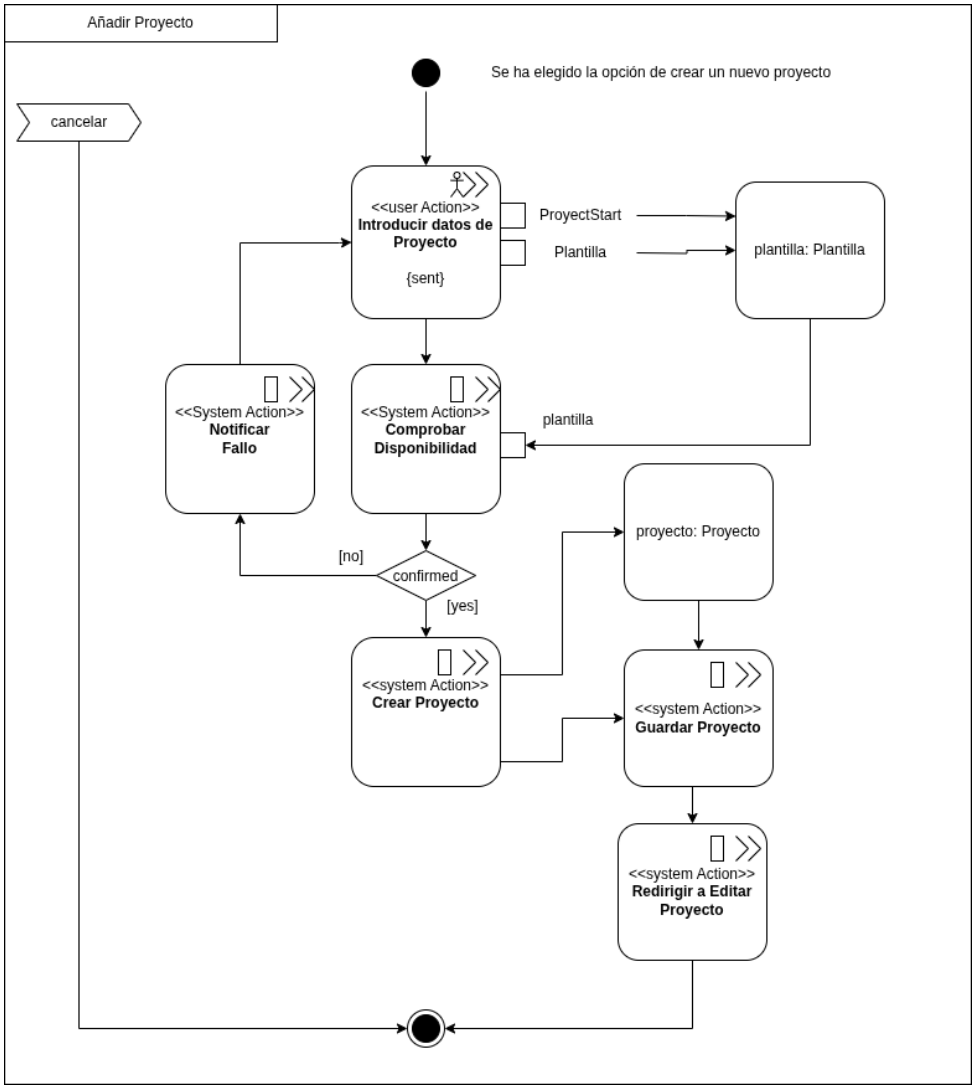

Figura 29: Diagrama de proceso - Crear Proyecto

En la figura29 se muestra el proceso para crear un nuevo proyecto hidratando una plantilla de proyectos. El administrador empieza eligiendo una plantilla de proyectos, tras lo cual se comprueba la disponibilidad de usuarios en la fecha elegida para llevar a cabo el proyecto, en caso de que no sea posible asignar usuarios se notificará del error al administrador, en otro caso se crea el proyecto y se guarda, tras lo cual se redirige al proceso para editar un proyecto ya creado.

7.5.9. Diagrama de Proceso - Editar Proyecto

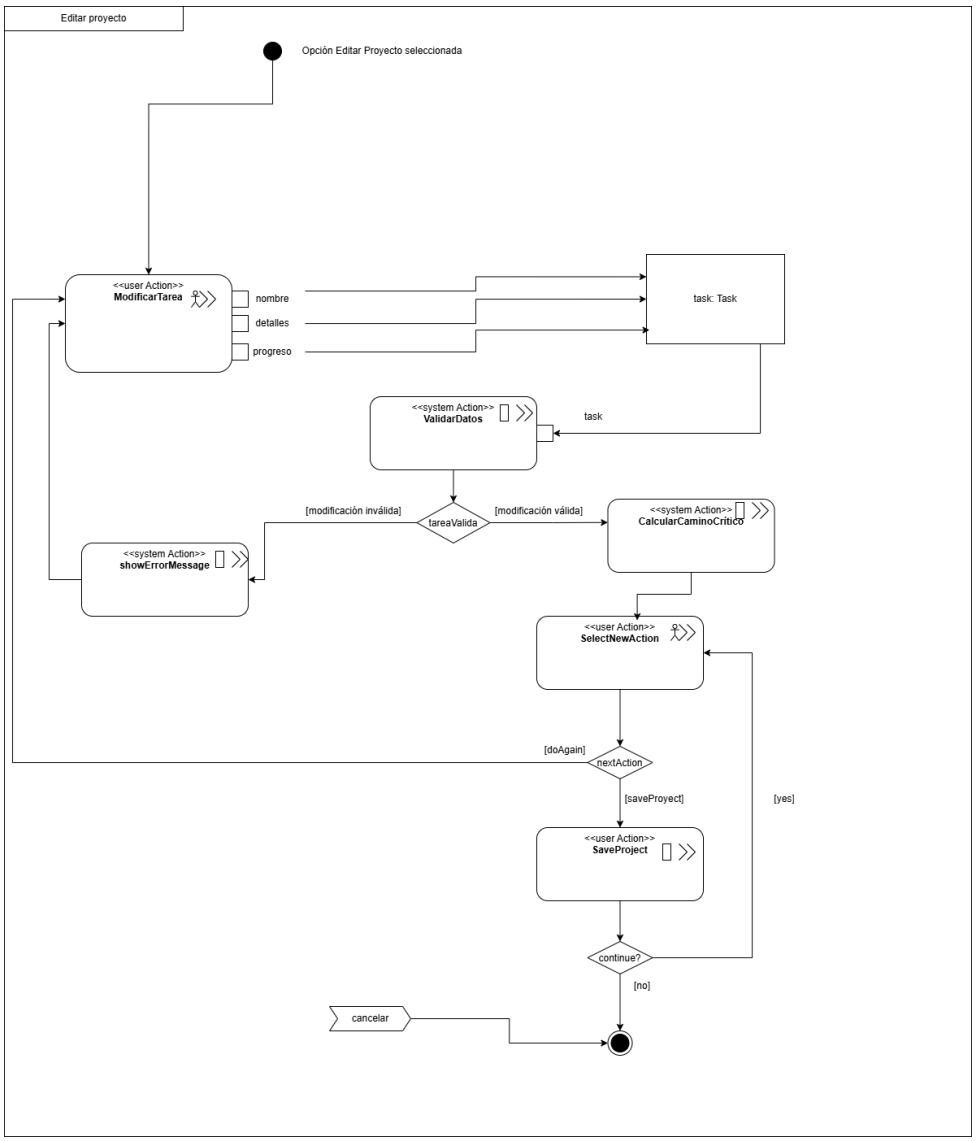

Figura 30: Diagrama de Proceso - Editar Proyecto

En la figura30 se muestran las acciones que puede realizar el administrador sobre el proyecto, en este caso solo la modificación de ciertos datos no esenciales, tras lo cual se comprueba si la modificación es válida, si lo es el administrador puede elegir realizar otra acción o guardar los cambios, en cualquier momento se puede cancelar todo el proceso.

7.5.10. Diagrama de Proceso - Actualizar proyectos

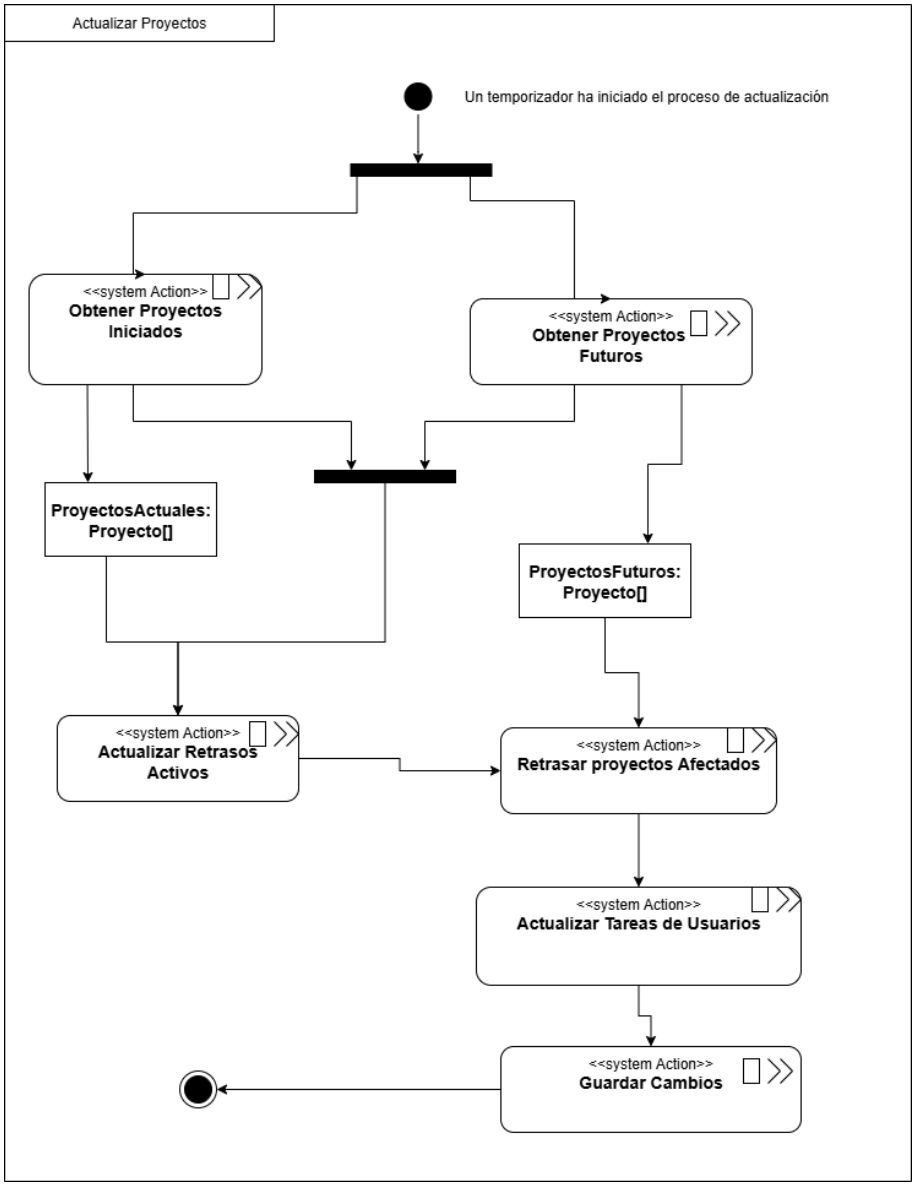

Figura 31: Diagrama de Proceso - Actualizar proyectos

La figura31 muestra el proceso que realiza el sistema para actualizar la duración de proyectos y retraso de los proyectos. se ejecuta periódicamente este proceso, en el que se obtienen los proyectos que actualmente (por fechas) se tengan que estar realizando en este momento, ademas de los proyectos que aun no han empezado, se actualizan las tareas de los proyectos si se han retrasado, y se procede a propagar esos cambios en caso de que afecten a otros proyectos, retrasando su fecha de inicio, tras eso se actualizan las tareas que los usuarios tienen asignadas y se guardan los cambios.

8. ARQUITECTURA DEL SISTEMA

Para el desarrollo de esta aplicación se han utilizado tecnologías consideradas actuales en el entorno web, dibujando una linea clara de separación entre cliente y servidor para gestionar independientemente las partes de la aplicación ademas de facilitar el mantenimiento de la misma.

El cliente ha sido implementado en Angular 2, el servidor en NodeJS haciendo uso del framework ExpressJS para facilitar tareas como configuración de conexiones a bases de datos y escucha de peticiones de diferentes origines, La base de datos está implementada mediante MongoDB.

Esta combinación forma el *tech-stack MEAN*, acrónimo de todas estas tecnologías antes mencionadas y cuyo motivo de elección para el desarrollo de esta aplicación web será explicado en la tabla18.

Tecnología	**Motivación**
Angular	Genera aplicaciones web dinámicas y de buen rendimiento, altamente escalables y faciles de mantener, ya que permite desplegar actualizaciones calientes del software, ademas permite integrar fácilmente servicios RESTful[13], Ademas el uso de Typescript permite escribir código mas robusto y relativamente fácil de depurar.
NodeJS	Permite ejecutar javascript en el lado del servidor[15], permitiendo la creación de APIs rápidamente al usar en el servidor un lenguaje similar al del cliente, ademas permite desplegar servidores de forma sencilla y con pocas lineas, disminuyendo la complejidad en gran medida.
ExpressJS	Debido a que es un Framework minimalista orientado a crear APIs RESTful [16], disminuyendo la complejidad de configurarlas y crear servicios.
MongoDB	Es una base de datos NoSQL Muy util debido a que ya sirve la información en JSON, fácil de tratar mediante javascript y Typescript, ademas de es rápido construir con el una base de datos, ademas,El acceso a los datos mediante operaciones CRUD es muy Rápido. y está ideado para ser escalable.[17]

Tabla 18: Tecnologías elegidas para la implementación del sistema

Con las tecnologías usadas en el proyecto, se pueden describir los principales componentes del sistema desarrollado, exponiendo primero la jerarquía de directorios del proyecto para fácilitar la ubicación de los elementos y funcionalidades del sistema. A continuación se muestra exactamente en la figura32 la estructura que existe en el repositorio.

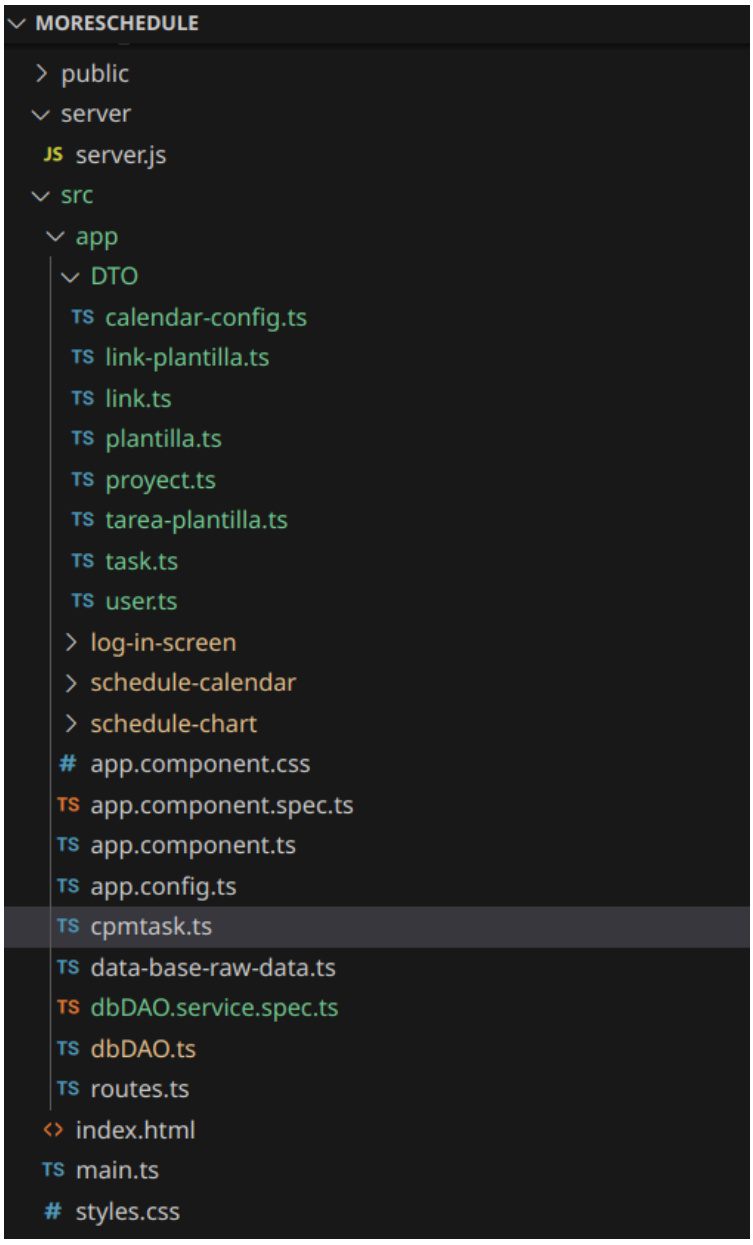

Figura 32: Jerarquía de directorios del cliente (`src/app`)

En la Figura32 puede observarse lo siguiente (solo se incluye la parte de `src/app` correspondiente al cliente Angular, ya que el resto de la jerarquía del proyecto no ha variado):

- `src/app/DTO/`: Carpeta que agrupa objetos de transferencia de datos (DTOs). Contiene los ficheros:
  - `calendar-config.ts`
  - `link-plantilla.ts`
  - `link.ts`
  - `plantilla.ts`
  - `proyect.ts`

    - tarea-plantilla.ts
    - task.ts
    - user.ts

- src/app/log-in-screen/: Carpeta con los componentes, plantillas y estilos propios de la pantalla de inicio de sesión.
- src/app/schedule-calendar/: Carpeta que contiene la implementación de la vista de calendario interactivo.
- src/app/schedule-chart/: Carpeta donde se encuentran los componentes y archivos necesarios para el gráfico de programación.
- En el nivel raíz de src/app/ (además de las carpetas anteriores) existen estos ficheros principales:

    - app.component.css
    - app.component.spec.ts
    - app.component.ts
    - app.config.ts
    - cpmtask.ts
    - dbDAO.service.spec.ts
    - dbDAO.ts
    - routes.ts
    - index.html
    - main.ts
    - styles.css

Con esto, queda reflejada con precisión la organización de la parte cliente de la aplicación. A partir de aquí, describiré el contenido y la función de cada uno de esos bloques, explicando módulos externos utilizados, ademas de funciones de interés para la comprensión de las responsabilidades del frontend en cada módulo.

8.1. Cliente

Todo lo que reside bajo `src/app/` conforma la aplicación web del lado del cliente. A continuación se describen las carpetas y ficheros tal como aparecen en la jerarquía, explicando brevemente su propósito:

- `DTO/`: Aquí se definen los *Data Transfer Objects* que se utilizan para tipar la comunicación con el servidor. Cada archivo coincide con un modelo de datos que la aplicación necesita manejar:
  - `calendar-config.ts`: Define la estructura de datos para la configuración del calendario laboral en el cliente.
  - `link-plantilla.ts` y `link.ts`: Tipan los enlaces entre tareas y plantillas (dependencias).
  - `plantilla.ts`: Interfaz que describe los campos de una plantilla.
  - `proyect.ts`: Interfaz con los datos de un proyecto.
  - `tarea-plantilla.ts`: Modelo para las tareas pertenecientes a una plantilla.
  - `task.ts`: DTO que tipa la información de una tarea en el contexto de un proyecto.
  - `user.ts`: Define el objeto con los atributos que recibe y envía el cliente sobre usuarios.
- `log-in-screen/`: Carpeta con todos los archivos necesarios para la pantalla de inicio de sesión. Contiene:
  - Componentes Angular (fichero `.ts`), sus plantillas (`.html` empotrado en el .ts) y estilos (`.css`) específicos para el formulario de login.
  - Toda la lógica de validación de usuarios, desde expresiones regulares hasta control de acceso de usuarios permitidos
  - gestiona la creación de cookies de sesión para mantener a un usuario dentro de las vistas de interés de la aplicación en caso de iniciar sesión correctamente
- `schedule-calendar/`: Directorio dedicado a la vista de calendario interactivo. En él se ubican:
  - Componente que generan y muestran el calendario, `schedule-calendar.component.ts`, `.html` (empotrado dentro del código .ts) y `.css`).
  - El elemento principal para la generación del calendario es un componente externo llamado *angular-calendar* desarrollado por Mattlewis92 como un proyecto de código abierto para mostrar calendarios y eventos en ellos[24] del estilo de *Google Calendar* o los calendarios de *Microsoft Teams*, lo cual se ajusta a la intención inicial del proyecto de mostrar todas las tareas en una vista global sencilla y fácilmente entendible.

- Lógica para marcar rangos de fechas, eventos, configuración visual, configuración del horario de trabajo, etc.
- En esta vista se ubica ademas la barra de menú principal de la aplicación. que nos facilita un acceso gráfico a formularios que permiten crear nuevos usuarios, editarlos, acceder al editor de plantillas y al visualizador de proyectos, configurar el horario de trabajo y por supuesto crear nuevos proyectos para una fecha respetando el horario de trabajo y eliminarlos si se cumplen las condiciones.
- todos estos formularios tendrán la forma de ventanas modales que se superponen a la vista principal del calendario para ofrecer una navegación poco recargada y fluida

- `schedule-chart/`: Carpeta que alberga los elementos necesarios para dibujar el gráfico de programación (Gantt, barras de tareas, etc.). Incluye:
  - Componentes específicos (`schedule-chart.component.ts`, `.html` (empotrado en el código .ts) y `.css`).
  - utilidades para graficar escalas temporales, cálculo del camino crítico, etc.
  - se utiliza otro componente externo, *DHTMLX Gantt*, cuya versión estándar es open source y distribuida bajo licencia publica de GNU. Es una librería muy completa para javascript y typescript para el graficado de tablas Gantt, ofreciendo utilidades para configurar la tabla y las tareas dentro de ella con un alto grado de flexibilidad y personalización[25], ofreciendo una librería software potente para gestión y planificación de proyectos incluso en su *versión standard*.

- Ficheros en `src/app/` (nivel raíz):
  - `app.component.ts`, `app.component.css` y `app.component.spec.ts`:
    - `app.component.ts` es el componente raíz (`AppComponent`). Contiene la lógica para arrancar la aplicación, inyecta servicios globales y define el `<router-outlet>` donde se cargan las vistas hijas.
    - `app.component.css` incluye estilos que afectan a la estructura global de todas las vistas, aunque en este caso no contendrá nada ya que serán las vistas la que se encarguen completamente de sus estilos.
    - `app.component.spec.ts` ofrece pruebas unitarias básicas para garantizar la creación y el comportamiento inicial de `AppComponent`.
  - `app.config.ts`: Archivo de constantes de configuración, utilizado para:
    - Aportar proveedores para los dististos servicios de la aplicación como el servicio de enrutado o establecer la coalescencia de eventos.
    - Mantener valores que se usan en distintos servicios y componentes.

- un ejemplo de esta configuración puede ser como el mostrado en la tabla19

```
export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routeConfig),
    provideNoopAnimations()
  ]
};
```

Tabla 19: Configuración de appConfig

- `dbDAO.ts` y `dbDAO.service.spec.ts`:
  - `dbDAO.ts` implementa el *Data Access Object* que gestiona todas las llamadas HTTP a la API del servidor, proporcionando operaciones CRUD importantes para la administración y tratamiento de los datos de la aplicación. Algunos de los métodos implementados aqui son:
    - `getAllUsers()`, `createUser()`, `updateUser()`, `deleteUser()`.
    - `getAllProjects()`, `getProjectById()`, etc.
    - `getTemplateById()`, y demás para tareas y enlaces. (Se explicará en mas detalle en la subsección de comunicación
  - `dbDAO.service.spec.ts` contiene las pruebas unitarias correspondientes a `dbDAO.ts`.
- `routes.ts`: Archivo donde se configuran las rutas internas de la aplicación Angular. Incluye:
  - Rutas principales como " (página de login), `'Calendar'`, `'proyectSchedule'`
- `index.html`: Página raíz que contiene la etiqueta `<app-root>`. Angular inyecta aquí la aplicación compilada y los scripts resultantes del build.
- `main.ts`: Archivo que arranca el motor de Angular y llama a `bootstrapApplication(AppComponent)`. Se encarga de:
  - Importar `AppComponent`.
  - Manejar errores en el arranque (con `catch((err) =>console.error(err))`).
  - Configurar proveedores globales.
- `styles.css`: Hoja de estilos global que se aplica a toda la SPA. Se usan aquí:
  - Reglas básicas de presentacion(`* { margin: 0; }`) para garantizar consistencia entre navegadores.

Con esta descripción, queda fielmente reflejada la organización de la parte cliente de la aplicación, Queda claro que la aplicación está modularizada, para que cada vista esté aislada del resto y funcione solo con la información requerida por ella, y explicando fielmente las bases de la configuración de los componentes.

8.2. Servidor

El servidor se encuentra dentro de la carpeta `server` y tiene dos responsabilidades, la primera es exponer una API RESTful que permita al cliente leer y escribir datos en la base de datos MongoDB, La otra responsabilidad es actualizar periodicamente el estado de los proyectos y sus tareas en la base de datos para tener información actualizada sobre el efecto de retrasos en de un proyecto en el resto de proyectos de la base de datos.

atendiendo a la primera responsabilidad, mediante NodeJS y ExpressJS el servidor expone los siguientes servicios:

- **Conexión a MongoDB:**
  - Al arrancar (`main()`), el servidor crea un `MongoClient` apuntando a `mongodb://localhost` y selecciona la base de datos `MoreScheduleDBdata`.
  - Si la conexión se establece correctamente, muestra en consola el mensaje de conexión:

    `Conectado a MongoDB: mongodb://localhost:27017 → MoreScheduleDBdata`
  - En caso de fallo en la conexión, interrumpe la ejecución y muestra el error.

```
const client = new MongoClient(MONGO_URL);
await client.connect();
const db = client.db(DB_NAME);
console.log(‘Conectado a MongoDB: ${MONGO_URL} — ${DB_NAME}‘);
```

Tabla 20: Conexión a MongoDB mediante MongoClient

- **Configuración de ExpressJS:**
  - Se instancia la aplicación con `const app = express()`.
  - Se habilita `cors()` para permitir que el cliente Angular (u otra fuente) consuma la API sin restricciones de origen.
  - Se añade `express.json()` para que todas las peticiones con cuerpo en formato JSON sean parseadas automáticamente.

```
const app = express();
app.use(cors());
app.use(express.json());
```

Tabla 21: Configuración de Express con CORS y JSON

- **Gestión de Proyectos (`proyects`):**
  - `POST /proyect/create`: Inserta un nuevo documento en la colección `proyects`. El cliente debe enviar un objeto JSON con campos `id`, `start`, `end`, `title` y `color`. Si la inserción es exitosa, responde con `{ inserted: ObjectId }`; en caso de error, devuelve un `500`.
  - `GET /proyect?pid=pid` : Busca en `proyects` todos los documentos cuyo campo `id` coincide con `pid`. Si `pid` no es un número válido, responde con `500` o `400`. En caso de éxito, devuelve un arreglo con los proyectos encontrados.
  - `DELETE /proyect/delete?pid=pid` : Elimina un único documento de `proyects` cuyo campo `id` coincida con `pid`. Si `pid` no es válido, responde `400`. Devuelve `{ deletedCount: N }` indicando cuántos documentos se borraron (0 o 1).
  - `GET /proyects`: Devuelve todos los documentos almacenados en la colección `proyects` como un arreglo JSON. En caso de error, responde con `500`.
- **Gestión de Tareas (`tasks`):**
  - `GET /tasks?pid=pid` : Devuelve todas las tareas cuyo campo `pid` coincide con el parámetro `pid`. Si `pid` no es un número válido, responde `400`. Retorna un arreglo con los documentos encontrados o `500` en caso de error interno.
  - `DELETE /tasks?pid=pid` : Borra todas las tareas de la colección `tasks` cuyo campo `pid` coincide con el parámetro `pid`. Usa el middleware interno `parsePid` para validar y convertir `pid`. Devuelve `{ deletedCount: N }` con el número de documentos eliminados.
  - `POST /tasks/batch`: Inserta un arreglo de tareas en lote. El cuerpo de la petición debe ser un array no vacío donde cada elemento contenga los campos obligatorios: `pid` (número), `id` (número), `text` (cadena), `start_date` (cadena) y `duration` (número). Si alguno de los objetos no cumple la tipificación, responde `400`. Si la inserción se completa, devuelve `{ insertedCount: M, insertedIds: [...] }`; en caso de error, `500`.
- **Gestión de Enlaces de Tareas (`links`):**
  - `GET /links?pid=pid` : Obtiene todos los enlaces de tareas para un proyecto dado. El parámetro `pid` debe ser un número válido; si no lo es, devuelve `400`. Retorna un arreglo con los documentos de `links` cuyo campo `pid` coincida.
  - `DELETE /links?pid=pid` : Elimina todos los documentos de `links` cuyo campo `pid` coincide con el parámetro. Utiliza el middleware `parsePid`, y devuelve `{ deletedCount: N }`.
  - `POST /links/batch`: Inserta varios enlaces en lote. Cada objeto del array de petición debe incluir `pid`, `id`, `source`, `target` (todos números) y `type` (cadena). Si falta o no es válido algún campo, responde `400`. En caso de éxito, devuelve `{ insertedCount: K, insertedIds: [...] }`; en caso contrario, `500`.

- **Gestión de Usuarios (`users`):**
  - `GET /users`: Devuelve todos los documentos almacenados en la colección `users`. En caso de fallo de lectura, responde `500`.
  - `GET /users/auth?uname=uname &pass=pass` : Busca un usuario que coincida con el nombre de usuario (`uname`, convertido a minúsculas y recortado) y la contraseña (`pass`, recortada). Si no se proporcionan ambos parámetros, responde `400`. Si no existe coincidencia, responde `404`. Si se encuentra, elimina la propiedad `pass` del objeto y devuelve el usuario como JSON. En caso de error interno, responde `500`.
- **Gestión de Calendario (`calendarConfig`):**
  - `GET /calendar/config`: Busca el primer documento en la colección `calendarConfig` y lo devuelve como objeto JSON (código `200`). Si hay algún error, responde `500`.
  - `POST /calendar/config/update`: Recibe en el cuerpo de la petición un objeto JSON con la nueva configuración de calendario. Primero elimina todos los documentos existentes en `calendarConfig` y, a continuación, guarda el objeto recibido como único documento. Si la operación es exitosa, responde `201` con `{ insertedId: ObjectId }`; en caso de error, `500`.
- **Gestión de Plantillas (`Templates`, `TemplateTasks`, `TemplateLinks`):**
  - `POST /templates/create`: Inserta un nuevo documento en la colección `Templates` con los campos `id`, `end` y `title` obtenidos del cuerpo. Si la inserción es correcta, responde con `200` y el JSON `{ inserted: ObjectId }`; en caso de error, `500`.
  - `GET /templates`: Devuelve todas las plantillas almacenadas en `Templates` como un arreglo JSON. Si ocurre un error al leer, responde `500`.
  - `GET /template?tid=tid` : Busca en `Templates` los documentos cuyo campo `id` coincide con `tid`. Si `tid` no es un número válido, responde `400`. Devuelve un arreglo con los resultados; si hay un error interno, `500`.
  - `DELETE /template/delete?tid=tid` : Elimina un único documento de `Templates` cuyo campo `id` coincide con `tid`. Si `tid` no es válido, responde `400`. Retorna el JSON `{ deletedCount: N }` con la cantidad de documentos eliminados.
  - `GET /template/tasks?tid=tid` : Obtiene todas las tareas de plantilla (`TemplateTasks`) cuyo campo `tid` coincide con `tid`. Si `tid` no es un número válido, responde `400`. Devuelve un arreglo con las tareas; en caso de fallo, `500`.
  - `GET /template/links?tid=tid` : Devuelve todos los enlaces de plantilla (`TemplateLinks`) cuyo campo `tid` coincide con `tid`. Si `tid` no es válido, responde `400`. Si ocurre un error interno, responde `500`.
  - `DELETE /template/tasks?tid=tid` : Elimina todas las tareas de `TemplateTasks` cuyo campo `tid` coincide con `tid`. Si `tid` no es un número, responde `400`. Retorna `{ deletedCount: N }`.
  - `DELETE /template/links?tid=tid` : Borra todos los enlaces de `TemplateLinks` cuyo campo `tid` coincide con `tid`. Si `tid` no es válido, responde `400`. Retorna `{ deletedCount: N }`.

- `POST /template/tasks/batch`: Inserta en lote varias tareas de plantilla. El cuerpo debe ser un array no vacío de objetos que incluyan los campos `tid`, `id`, `text`, `start_date` y `duration`. Si falta alguno o no cumple el formato, responde `400`. En caso de éxito, devuelve `{ insertedCount: M, insertedIds: [...] }`; si falla, `500`.
- `POST /template/links/batch`: Inserta en lote varios enlaces de plantilla. Cada objeto del array debe contener `tid`, `id`, `source`, `target` y `type`. Si algún campo es inválido o falta, responde `400`. Si la operación se completa, retorna `{ insertedCount: K, insertedIds: [...] }`; de lo contrario, `500`.

De esta forma, el servidor cubre todas las operaciones CRUD necesarias para gestionar proyectos, tareas, enlaces, usuarios, configuraciones de calendario y plantillas (tanto datos individuales como inserciones en lote). Cada ruta está implementada directamente en `server.js` sin archivos intermedios adicionales, de modo que la lógica de cada endpoint puede revisarse fácilmente en un único fichero.

A nivel de arquitectura, se delega toda la lógica de validación básica (comprobar que parámetros `pid` o `tid` sean números, que los arrays de inserción no estén vacíos, etc.) a condiciones en cada endpoint. Esto simplifica la estructura del servidor, concentrando tanto la configuración de Express como la definición de rutas en un mismo punto. Así, el cliente conoce exactamente qué URIs debe invocar y qué formato de datos debe enviar o esperar como respuesta.

Ahora, con respecto a la segunda responsabilidad. El servidor se encargará de actualizar periódicamente el estado de las tareas como se muestra en la tabla22:

```
    setInterval( async() =>{
    try{
        await checkChangeOnCalendarProyectDuration()
    }catch (error){
        console.error('UNEXPECTED ERROR ON EXECUTION ADJUSTMENT
            EXECUTION', error);
    }
},MINUTES_INTERVAL * 60000)


checkChangeOnCalendarProyectDuration();
```

Tabla 22: Actualización del estado de los proyectos

Cada cierto tiempo se volverá a entrar en el bloque de este intervalo, calculando que proyectos están activos a fecha de la ejecución de la función (cuales han empezado), y como afectan sus retrasos a los proyectos que están en el futuro (cuales aún no han empezado), atendiendo a criterios como la holgura de una tarea, que en el momento de la ejecución, aun no haya sido marcada como terminada y cuanto retraso lleva una tarea dentro de un proyecto.

Externamente se puede resumir la lógica de dicha actualización de estados en:

1. Calcular el estado de las tareas de los proyectos activos
2. Propagar los cambios al resto de proyectos futuros que dependan en alguna medida del proyecto activo actual
3. Actualizar los datos en la base de datos de la aplicación

En total, el servidor tiene lo necesario para poder encargarse tanto de la comunicación con la base de datos como de la actualización continua de los datos de interés de la aplicación.

8.3. Base de Datos

La base de datos empleada es MongoDB, almacenando la información en varias colecciones que guardan documentos en formato JSON (BSON internamente). A continuación se describen las colecciones reales que existen en `MoreScheduleDBdata`, junto con los campos que contienen y un breve ejemplo de sus documentos tal como se muestran en MongoDB Compass.

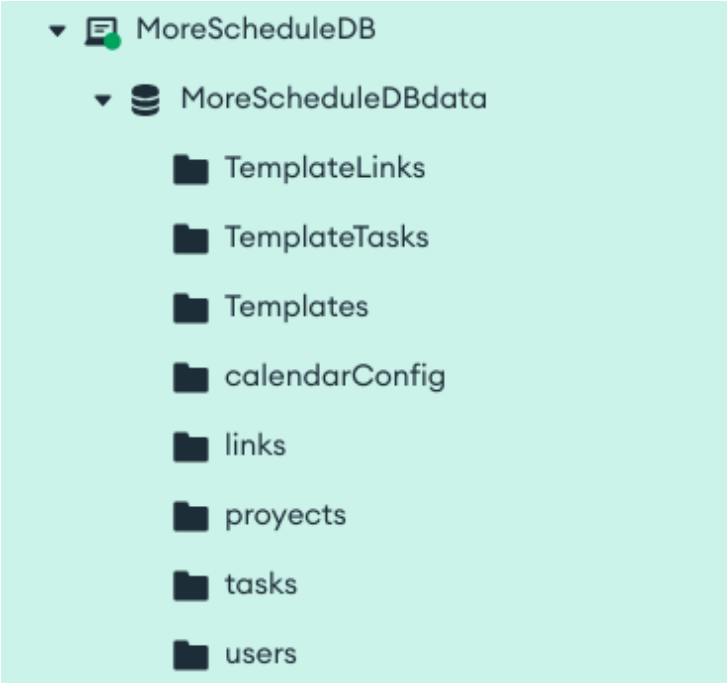

Figura 33: Vista general de las colecciones en MongoDB

- **Colección `TemplateLinks`**
  Esta colección almacena los enlaces (dependencias) entre tareas dentro de una misma plantilla. Como puede verse en la tabla23, cada documento contiene los campos siguientes:

  - `_id`: ObjectId generado automáticamente por MongoDB.
  - `tid`: Identificador numérico de la plantilla a la que pertenecen estos enlaces.
  - `id`: Identificador numérico único de este enlace dentro de la plantilla.
  - `source`: Identificador numérico de la tarea de origen.
  - `target`: Identificador numérico de la tarea de destino.
  - `type`: Cadena que indica el tipo de dependencia (por ejemplo, `"0"` para *finish-to-start*).

```
{
  "_id"   : ObjectId("683d58220221d9deb8d425a9"),
  "tid"   : 7035085,
  "id"    : 1748849907890,
  "source": 1748849907870,
  "target": 1748849907877,
  "type"  : "0"
}
```

Tabla 23: Ejemplo de documento en TemplateLinks

- **Colección** `TemplateTasks`
  En esta colección se guardan las tareas que forman parte de cada plantilla. Como puede verse en la tabla24, cada documento incluye:
  - `_id`: ObjectId asignado por MongoDB.
  - `tid`: Identificador numérico de la plantilla.
  - `id`: Identificador numérico **único** de la tarea dentro de la plantilla.
  - `text`: Texto descriptivo de la tarea (por ejemplo, `"montaje de la cubeta"`).
  - `start_date`: Fecha o instante de inicio, expresado como número decimal (por ejemplo, `1.25`).
  - `duration`: Duración estimada de la tarea en unidades de tiempo (por ejemplo, `204`).
  - `user_count`: Cantidad de usuarios necesarios para poder iniciar la tarea (por ejemplo, `3`).

```
{
  "_id"       : ObjectId("683d58220221d9deb8d425a6"),
  "tid"       : 7035085,
  "id"        : 1748849907857,
  "text"      : "montaje de la cubeta",
  "start_date": 1.25,
  "duration"  : 204,
  "user_count": 2
}
```

Tabla 24: Ejemplo de documento en TemplateTasks

- **Colección** `Templates`
  Aquí se almacenan las plantillas. Como puede verse en la tabla25, cada Cada documento contiene:
  - `_id`: ObjectId generado automáticamente.
  - `id`: Identificador numérico **único** de la plantilla.
  - `end`: Fecha o instante simbólico de fin de plantilla (por ejemplo, `10`).
  - `title`: Nombre de la plantilla (por ejemplo, `"vareadora"`).

```
{
  "_id"  : ObjectId("683d5f2f0221d9deb8d425ac"),
  "id"   : 4171773,
  "end"  : 88,
  "title": "Nueva Plantilla"
}
```

Tabla 25: Ejemplo de documento en Templates

- **Colección** `calendarConfig`
  Esta colección contiene la configuración general del calendario. Como puede verse en la tabla26, Solo hay un documento que agrupa:
  - `_id`: ObjectId generado por MongoDB.
  - `entrada`: Hora de inicio de la jornada (por ejemplo, `"08:00"`).
  - `salida`: Hora de fin de la jornada (por ejemplo, `"18:00"`).

```
{
  "_id"     : "6834498fc2cd88a3e5bfafe9",
  "entrada" : "08:00",
  "salida"  : "18:00",
  festivos  : []
}
```

Tabla 26: Ejemplo de documento en calendarConfig

- **Colección** `links`
  La colección `links` almacena los enlaces de dependencias entre tareas dentro de un proyecto activo.Como puede verse en la tabla27, los campos de cada documento serán:
  - `_id`: ObjectId de MongoDB.
  - `pid`: Identificador numérico del proyecto al que pertenece el enlace.
  - `id`: Identificador numérico **único** de este enlace dentro del proyecto.
  - `source`: Identificador numérico de la tarea de origen.
  - `target`: Identificador numérico de la tarea destino.
  - `type`: Cadena que establece el tipo de dependencia (por ejemplo, `"0"` especifica dependencia de fin a inicio).

```
{
  "_id"   : ObjectId("683d763d0221d9deb8d425be"),
  "pid"   : 51774640,
  "id"    : 1748849907890,
  "source": 1748849907870,
  "target": 1748849907877,
  "type"  : "0"
}
```

Tabla 27: Ejemplo de documento en links

- **Colección** `proyects`
  La colección `proyects` contiene los proyectos almacenados. Como puede verse en la tabla28, cada documento incluye:

  - `_id`: ObjectId asignado automáticamente.
  - `id`: Identificador numérico **único** del proyecto.
  - `start`: Fecha y hora de inicio del proyecto como cadena (por ejemplo, `"06-02-2025 12:00"`).
  - `end`: Duración estimada en horas (por ejemplo, `10`).
  - `title`: Nombre o título del proyecto (por ejemplo, `"Nuevo Proyecto"`).
  - `color`: Objeto que define los colores que se representarán en el calendario para identificar el proyecto (por ejemplo, `{"primary": "#ff0000", "secundary": "#ff6`

```
{
  "_id"   : ObjectId("683b044c107b21293e68c193"),
  "id"    : 15235454,
  "start": "01-01-1970 01:00",
  "end"   : 5,
  "title": "proyectName",
  "color": {
    "primary": "#00ff00",
    "secundary": "#006600"
  }
}
```

Tabla 28: Ejemplo de documento en proyects

- **Colección `tasks`**
  En la colección `tasks` se almacenan las tareas de cada proyecto. Como puede verse en la tabla29, cada documento contendrá:

  - `_id`: ObjectId de MongoDB.
  - `pid`: Identificador numérico del proyecto al que pertenece la tarea.
  - `id`: Identificador numérico **único** de la tarea dentro del proyecto.
  - `text`: Descripción de la tarea (por ejemplo, `"montaje de la cubeta"`).
  - `start_date`: Fecha y hora de inicio en formato cadena (por ejemplo, `"2025-06-02 13:15"`).
  - `duration`: Duración en unidades de tiempo (por ejemplo, `204`).
  - `offtime`: Tiempo que pasa la tarea fuera de jornada laboral, `104`).
  - `details`: comentarios de usuario en la tarea (por ejemplo, `se ha roto el manguito en el montaje`).
  - `slack`: Holgura de tiempo que se puede retrasar una tarea antes de suponer un retraso para el proyecto y futuros proyectos (por ejemplo, `43`).
  - `slack_used`: tiempo tras la fecha de finalización de la tarea que aun no ha acabado (por ejemplo, `22`).
  - `users`: usuarios asignados a una tarea (por ejemplo, `[{"uname": "pedro"}]`).

```
{
  "_id"       : ObjectId("683d763d0221d9deb8d425bb"),
  "pid"       : 51774640,
  "id"        : 1748849907857,
  "text"      : "montaje de la cubeta",
  "start_date": "2025-06-02 13:15",
  "duration"  : 204,
  "offtime" : 120,
  "details": "no ha llegado la chapa de 5",
  "slack": 23,
  "slack_used": 12,
  "users":[{
      "uname": "jose"
  }]
}
```

Tabla 29: Ejemplo de documento en tasks

- **Colección** `users`
  La colección `users` almacena los usuarios dentro de la aplicación y marcando sus límites y roles. Como puede verse en la tabla30, cada documento incluye:

  - `_id`: ObjectId generado por MongoDB.
  - `uname`: Nombre de usuario **único** (por ejemplo, `"pablo"`).
  - `pass`: Contraseña en texto plano (por ejemplo, `"Pablo123"`).
  - `admin`: Booleano que determina si el usuario es administrador (`true`) o un operario (`false`), en el segundo caso no se permite el login.
  - `Disponible`: variable que indica si el usuario puede ser contemplado o no para la asignación de tareas (por ejemplo, `false`).
  - `tareas`: pila de tareas asignadas al usuario, el primer indice contiene la tarea asignada mas recientemente (por ejemplo, `[ "tarea": .ª", .ªcaba": "2025-07-25 10:57", "pid": 94437947, "tid": 1752524271104 ]`).

```
  {
  "_id": ObjectId("683d763d0221d9deb8d425bb"),
  "uname": "alejandro",
  "pass": "Alejandro123",
  "admin": true,
  "disponible": true,
  "tareas": [
    {
      "tarea": "a",
      "acaba": "2025-07-28 08:47",
      "pid": 75046695,
      "tid": 1752524271104
    },
    {
      "tarea": "a",
      "acaba": "2025-07-25 10:57",
      "pid": 94437947,
      "tid": 1752524271104
    }
  ]
}
```

Tabla 30: Ejemplos de documentos en users

En conjunto, estas colecciones permiten gestionar:

- Las plantillas y sus dependencias (`TemplateLinks`, `TemplateTasks`, `Templates`).
- La configuración del calendario (`calendarConfig`).
- Los proyectos y sus tareas (`proyects`, `tasks`, `links`).
- La autenticación y autorización de usuarios (`users`).

Todos los ejemplos anteriores reflejan fielmente la estructura y los campos que aparecen en MongoDB Compass.

8.4. Comunicación

La comunicación en la aplicación se divide en dos niveles principales:

1. La interacción cliente–servidor, donde Angular consume la API RESTful expuesta por ExpressJS mediante el `dbDAO`.
2. El enrutamiento interno de la SPA, donde se navega entre vistas y se pasan datos mediante `queryParams`.

1. Interacción Cliente–Servidor Todos los accesos a datos se realizan a través de la clase `dbDAO`, inyectada con `HttpClient` o usando `fetch` directo para algunas llamadas específicas. A continuación se describen los métodos más relevantes y sus correspondientes endpoints:

- **Usuarios**
  - `GetUser(uname, pass)`:
    - Usa `http://localhost:3000/users/auth`.
    - Si la respuesta es `200`, retorna un objeto `User`. Si es `404`, retorna `undefined`.
- **Configuración de Calendario**
  - `GetCalendarConfig()`:
    - Llama a `"http://localhost:3000/calendar/config"` y parsea el JSON como `CalendarConfig`.
  - `updateCalendarConfigPromise(config)`:
    - Invoca `POST http://localhost:3000/calendar/config/update` con el objeto `config` en el cuerpo; retorna un `Promise<number>` con el `insertedId`.
- **Proyectos**
  - `GetProyect(Pid)`:
    - Accede a `http://localhost:3000/proyect`. Si la respuesta es `200`, devuelve un `Proyect`; en caso contrario, `undefined`.
  - `GetProyects()`:
    - Accede a `http://localhost:3000/proyects`. Convierte el arreglo de `Proyect` recibido en una lista de `CalendarEvent` para el calendario del cliente.
  - `createProyect(proyect)`:
    - Usa `POST http://localhost:3000/proyect/create` con un Proyect en el cuerpo; retorna el `Promise<void>` tras `lastValueFrom`.
  - `deleteProyectByPidPromise(pid)`:

- Llama a `DELETE http://localhost:3000/proyect/delete` y mapea `deletedCount` a `Promise<number>`.

- **Tareas de Proyecto**
  - `GetProyectTasks(id)`:
    - Accede a `http://localhost:3000/tasks)`, convierte cada documento a `Task` y retorna un `Promise<Task[]>`.
  - `createTasksBatch(tasks)`:
    - Envía `POST http://localhost:3000/tasks/batch` con un arreglo de `Task`. Retorna un `Observable<number>` con `insertedCount`.
  - `deleteTasksByPid(pid)`:
    - Ejecuta `DELETE http://localhost:3000/tasks` y, usando `pipe(map(res =>res.deletedCount))`, retorna el numero de tareas borradas.
- **Enlaces de Proyecto**
  - `GetProyectLinks(id)`:
    - Accede a `http://localhost:3000/links`, convierte a `Link[]` y devuelve una lista de enlaces.
  - `createLinksBatch(links)`:
    - Usa `POST http://localhost:3000/links/batch` con un arreglo de `Link`. Retorna `Observable<number>` como el numero de links insertados.
  - `deleteLinksByPid(pid)`:
    - Ejecuta `DELETE http://localhost:3000/links?pid=${pid}` y retorna `Observable<number>` con `deletedCount` como el número de enlaces borrados.
- **Plantillas**
  - `GetTemplates()`:
    - Llama a `http://localhost:3000/templates`, consigue todas las plantillas y las retorna.
  - `GetTemplate(Tid)`:
    - Realiza `http://localhost:3000/template`. Si `200`, devuelve `Plantilla`; en caso contrario, `undefined`.
  - `createTemplate(template)`:
    - Envía `POST http://localhost:3000/templates/create` con el objeto `Plantilla`. Retorna `Promise<void>`.
  - `deleteTemplateByTidPromise(tid)`:
    - Ejecuta `DELETE http://localhost:3000/template/delete` y retorna `Promise<numbe` con `deletedCount`.
  - `GetTemplateTasks(id)`:

    - Llama a `http://localhost:3000/template/tasks`, convierte a `TareaPlantilla[]` y devuelve `Promise<TareaPlantilla[]>` con la lista de tareas de la plantilla.
- `createTemplateTasksBatch(tasks)`:
    - Uso de `POST http://localhost:3000/template/tasks/batch` con un arreglo de `TareaPlantilla`. Retorna un `Observable<number>` con la cantidad de tareas introducidas.
- `deleteTemplateTasksByPid(tid)`:
    - Ejecuta `DELETE http://localhost:3000/template/tasks` y retorna `Observable<numb` el numero de tareas de la plantilla borradas.
- `GetTemplateLinks(id)`:
    - Realiza `http://localhost:3000/template/links`, convierte a `LinkPlantilla[]` y devuelve una promesa de lista de enlaces de plantilla.
- `createTemplateLinksBatch(links)`:
    - Envía `POST http://localhost:3000/template/links/batch` con un arreglo de `LinkPlantilla`. Retorna `Observable<number>` como el numero de enlaces de la plantilla introducidos.
- `deleteTemplateLinksByPid(tid)`:
    - Ejecuta `DELETE http://localhost:3000/template/links` y retorna `Observable<number>` como el numero de enlaces de la plantilla eliminada.

La mayoría de los métodos usan `fetch(...)` para lecturas puntuales (conversión manual a los DTO correspondientes) o `HttpClient` para las operaciones batch y de escritura, transformando los observables a promesas usando `lastValueFrom`. De esta forma, el cliente se abstrae de las rutas exactas del servidor y solo consume los métodos del `dbDAO`, que implementan toda la lógica de envío, la validación básica de parámetros (como `pid` o `tid`) y el parseo de la respuesta.

2. Enrutamiento de la SPA y paso de datos El enrutamiento interno se define en el fichero `routes.ts`, que exporta un arreglo de `Routes`. El fragmento relevante es el mostrado en la tabla31:

```
import { Routes } from "@angular/router";
import { ScheduleChartComponent } from "./schedule-chart/schedule-
    chart.component";
import { LogInScreenComponent } from "./log-in-screen/log-in-screen.
    component";om
import { ScheduleCalendarComponent } from "./schedule-calendar/
    schedule-calendar.component";

const routeConfig: Routes = [
  {
    path: '',
    component: LogInScreenComponent,
    title: 'Bienvenido a MoreSchedule'
  },
  {
    path: 'proyectSchdedule',
    component: ScheduleChartComponent,
    title: 'MoreScheduler'
  },
  {
    path: 'Calendar',
    component: ScheduleCalendarComponent,
    title: 'MoreScheduler'
  }
];

export default routeConfig;
```

Tabla 31: Fichero Routes.ts

Para navegar entre estas rutas y pasar datos (por ejemplo, el `pid` del proyecto seleccionado), se emplean `queryParams`. Un ejemplo en un componente que redirige a la vista de tabla Gantt sería:

```
// En el calendario, tras seleccionar un proyecto con id = proyecto.id
this.router.navigate(
  ['/proyectSchdedule'],
  {
    queryParams: { id: proyecto.id ,
    title: 'verProyecto',
    name: 'Proyecto 4'
    }
  }
);
```

Tabla 32: Routing de Calendario a proyectSchedule

Y en `ScheduleChartComponent` se recupera así el `pid` en `ngOnInit()`:

```
import { ActivatedRoute } from "@angular/router";

export class ScheduleChartComponent implements OnInit, AfterViewInit,
   OnDestroy {


  private dbDao: dbDAO = inject(dbDAO);
  private data: Task[] = [];
  private cpmTasks: CPMTask[] = [];
  private route: ActivatedRoute = inject(ActivatedRoute);
  private router: Router = inject(Router);
  private cookie: CookieService = inject(CookieService)
  //AQUI SE OBTIENEN LOS PARAMETROS DE LA URI
  private id = this.route.snapshot.queryParams['id'];
  private mode: string = this.route.snapshot.queryParams['title'];

  private timerID: number = 0;

  public nameContent: string = "";
  public saveButtonName: string = "";
```

Tabla 33: QueryParams en proyectSchdeule - verProyecto

De igual modo, al pasar de la vista de Gantt al calendario, se usa:

```
this.router.navigate(
  ['/Calendar']
);
```

Tabla 34: Routing de proyectSchedule a Calendario

Con este esquema de enrutado y paso de `queryParams`, la aplicación mantiene su estado de forma reactiva y permite al usuario compartir o recargar la URL sin perder el contexto, ya que el parámetro `pid` queda expuesto en la ruta. Asimismo, el `dbDAO` se encarga de intercambiar datos con el servidor según las necesidades de cada componente, garantizando la separación de responsabilidades entre lógica de negocio, acceso a datos y navegación de la SPA.

9. PRUEBAS, ERRORES Y SOLUCIONES

Durante el desarrollo de este documento se han discutido entre otras cosas, aspectos del diseño general del sistema. así como aspectos de la implementación de las diferentes partes de la aplicación. Ahora con los diferentes módulos y funcionalidades prometidas finalmente desarrolladas, y con un grado de desempeño suficiente para probar la funcionalidad de la app, se han desarrollado las siguientes pruebas (clasificadas por componente) para comprobar que el software se comporta de acuerdo a lo esperado.

9.1. LogIn Screen

Prueba	**Errores**	**Soluciones**
Inicio de sesión de usuarios con credenciales válidas	La página de inicio quedaba atascada en un bucle infinito de carga una vez se inicia sesión debido a lógica de redirección errónea. Una vez iniciada sesión se redirigía a la página del calendario, que tenía una lógica de redirección incorrecta si había una cookie de sesión, redirigiendo nuevamente a la página de inicio, que al detectar la cookie redirigía de nuevo al calendario.	Cambio en la cláusula 'if' que manejaba la existencia de la cookie.

Tabla 35: Inicio de sesión con credenciales válidas

Prueba	**Errores**	**Soluciones**
Inicio de sesión con usuarios sin credenciales válidas	Pasadas las pruebas exitosamente	No aplica

Tabla 36: Inicio de sesión sin credenciales válidas

Prueba	**Errores**	**Soluciones**
Acceso a páginas de la app sin iniciar sesión	La aplicación permitía acceder a vistas prohibidas para usuarios sin sesión activa, ya que las páginas no comprobaban la existencia de la cookie de sesión.	Añadida cláusula 'if' en los componentes que retornan a la vista de Login si no encuentran la cookie de sesión.

Tabla 37: Acceso a páginas de la app sin iniciar sesión

9.2. Schedule Calendar

Prueba	Errores	Soluciones
Asignación de tareas a usuarios	Al crear un proyecto eligiendo una fecha automáticamente, daría una fecha en la que no hay usuarios disponibles para el proyecto, bloqueando el flujo y no creando la tarea, pese a que el mecanismo nos da una fecha que pasa los criterios. el problema se debía a que la función que hace la asignación de usuarios recibía la referencia al objeto de dichos usuarios, por tanto asignaba la tarea al usuario cuando comprobaba que podía encargarse de ella	realicé la copia profunda del objeto que recibía la función, rompiendo el paso por referencia y no modificando los objetos reales

Tabla 38: Asignación de tareas a usuarios

Prueba	Errores	Soluciones
Creación de nuevo proyecto	Al crear un nuevo proyecto, el formato de fecha era diferente al que esperaba el componente de calendario, no mostrando el proyecto en el calendario aunque se guardase en la base de datos	cambio del formato de fecha en los proyectos de la base de datos *ƶyyy-MM-dd HH:mm*" por "*MM-dd-yyyy HH:mm*"
Creación de nuevo proyecto	Al crear un nuevo proyecto, las tareas no mantenían sus relaciones de dependencia, pudiendo empezar antes que sus predecesores, debido a que la función que hidrata una plantilla en un proyecto no encontraba la tarea de la que dependía aunque exisistiera un enlace entre las tareas	procedí a ordenar las tareas de menor a mayor fecha de inicio, asegurando que siempre se procesen los predecesores de una tarea antes de la misma
Creación de nuevo proyecto	Al crear un nuevo proyecto si seleccionamos la opción de elegir fecha automáticamente, y cambiábamos la plantilla que habíamos elegido, la fecha que se muestra para crear la plantilla es el **1 de enero de 1970**, debido a que el bloque if que detectaba el cambio estaba mal definido, dejando el valor de la plantilla como **undefined**	procedí a cambiar la clausula if else, detectando propiamente el cambio de plantilla de proyecto

Tabla 39: Creación de nuevo proyecto

Prueba	Errores	Soluciones
Eliminación de proyectos	Al eliminar un proyecto, las tareas del mismo asignadas a usuarios no se borrarían, debido a que la lógica de actualización de usuarios de la base de datos requería comparar la longitud del *array* de tareas asignadas a usuario en la interfaz y los usuarios que se usan como referencia para comparar con lo que debería haber en la base de datos, dichos arrays mantenían referencias a los mismos objetos, provocando que la comparación dijera que los arrays eran iguales.	procedí a realizar la copia profunda de dichos usuarios y sus tareas, creando dos arrays de objetos con diferentes referencias.

Tabla 40: Eliminación de proyectos

Prueba	Errores	Soluciones
Creación de usuarios	No se encontraron errores	No aplica

Tabla 41: Inicio de sesión sin credenciales válidas

9.3. Schedule Chart

Prueba	Errores	Soluciones
Configuración de tabla Gantt para Editar Plantilla	A la hora de configurar la caja de configuración de la tarea de una plantilla, se configuró para que tuviera un campo personalizado fuera de la especificación del objeto de tareas de la tabla Gantt, pese a que el campo se mostraba en el cajón de configuración, este no se guardaba en los datos de la tarea, por lo que al guardar las tareas de la plantilla en la base de datos se producía un error porque el valor del numero de usuarios no está definido	manejé el evento de creación de tareas de plantilla de forma que añadiese un campo de tipo *index signature* para la cantidad de usuarios de la tarea, el cual se añadía y ahora se actualizaba cada vez que cambiaba el elemento del *frontend* que lo manejaba, gracias a esto ese campo está definido desde la creación de la tarea.

Tabla 42: Configuración de tabla Gantt para Editar Plantilla

Prueba	Errores	Soluciones
Configuración de tabla Gantt para ver proyecto	El campo que muestra los usuarios asignados a la tarea mostraba el nombre del operario de 1 a N veces, incrementando cada vez que accedía a la tabla Gantt para mostrar el proyecto, debido a que en vez de iniciar la cadena de caracteres del nombre de usuario en el campo necesario, sumaba el nombre de usuario al contenido que ya hubiera	cambia el operador suma con asignación por el operador asignación, así ignoro lo que hubiera antes en el campo, solo mostrando los usuarios contenidos en la tarea.

Tabla 43: Configuración de tabla Gantt para ver proyecto

Prueba	Errores	Soluciones
Calculo de camino Crítico	El camino crítico de todas las tareas mostraba incoherencias debido a que consideraba que el inicio de todas las tareas sin predecesores era 0, sin embargo cada tarea sin predecesores tiene un tiempo de retraso hasta su inicio, el cual no estaba contemplado en el calculo original del camino crítico	Añadí el retardo de inicio de las tareas sin predecesores al calculo de *Earliest Start*, lo que ajustó correctamente el cálculo del camino critico del proyecto

Tabla 44: Calculo de camino Crítico

Prueba	Errores	Soluciones
Guardado de proyectos y plantillas	No se encontraron fallos	No aplica.

Tabla 45: Guardado de proyectos y plantillas

9.4. DbDAO

Este apartado, debido a las pruebas realizadas en los anteriores componentes, que suponen un uso extenso de métodos definidos en **DbDAO** para actualizar, modificar, crear y eliminar los datos usados por la aplicación en la base de datos, se puede argumentar que el correcto funcionamiento de dichos componentes que utilizan estas funciones son prueba suficiente de la robustez, coherencia y buenas prácticas y comportamiento esperado de los métodos implementados en DbDAO.

9.5. Server

En este apartado, podemos considerar igualmente que si los métodos de DbDAO acceden y proporcionan los datos necesarios a los componentes de la aplicación, los puntos finales (*endpoints)* que encontramos en Server cumplen adecuadamente con la funcionalidad esperada de ellos.

Por tanto, para mayor facilidad y claridad para entender la importancia de las pruebas en este componente, nos centraremos principalmente en los métodos del servidor que se encargan de mantener la coherencia temporal y evitar solapamientos en las tareas conforme avanza el tiempo:

Prueba	**Errores**	**Soluciones**
Actualización de duración de tareas en diferentes fechas	Error de lógica en la función *ajustarTiempoDeFin* restaba dos veces el tiempo extra que el proyecto adquiere si se retrasa mas allá de su fecha de finalización, provocando que la duración del proyecto estuviera inflada. atrasando el resto de proyectos un tiempo innecesario	Eliminé la linea de código que sumaba dos veces la holgura (*slack*), haciendo que los cálculos volvieran a ser consistentes
Actualización de duración de tareas en diferentes fechas	Error de lógica en la función getUsuariosConUltimaTareaEnProyecto, recorría mal la pila de tareas asignadas al usuario, desde el final al principio, por lo que en vez de obtener la ultima tarea de un usuario en un proyecto, obtenía la primera, rompiendo toda la lógica que establezco para saber si la ultima tarea de un usuario en un proyecto afecta a la primera tarea de otro proyecto	invertí el orden en el que se recorría la pila. empezando en el indice 0 y bajando, lo cual encontraba ya la ultima tarea de un usuario en un proyecto dado.

Tabla 46: Actualización de duración de tareas en diferentes fechas

10. CONCLUSIONES

10.1. Cumplimiento de objetivos

A lo largo de este Trabajo Fin de Grado se han logrado los puntos esenciales marcados al inicio. Se ha puesto en marcha una SPA basada en el tech-stack MEAN (Angular, Node.js, Express, MongoDB), con una arquitectura modular sencilla de extender. El sistema de autenticación distingue administradores y operarios, y controla la disponibilidad antes de asignar tareas. Las plantillas de proyecto pueden crearse, editarse y guardarse, incluyendo sus tareas y dependencias. A partir de esas plantillas, se generan proyectos reales con asignación automática de personal y cálculo de fechas según el calendario laboral. La interfaz ofrece un calendario interactivo y diagramas de Gantt con visualización y actualización camino crítico, que permiten seguir retrasos y estimar entregas. El servidor actualiza de forma periódica los estados de los proyectos, propagando cualquier retraso entre los que aún no han comenzado. Se han validado todos estos flujos en un entorno empresarial real, completando pruebas funcionales en cada módulo.

10.2. Mejoras futuras

Para enriquecer la aplicación y adaptarla a nuevos escenarios, se podrían plantear distintas líneas de trabajo, introduciendo mejoras de calidad de vida y permitiendo la expansión de su uso fuera de MORESIL SL. Entre ellas, la que quizás sea de mayor interés sea la habilitación de soporte para múltiples empresas, aislando calendarios y datos de cada compañía, la inclusión de campos opcionales en las plantillas, como costes, materiales o prioridades, en función del tipo de proyecto; la incorporación de notificaciones por correo y alertas, junto con un sistema de control de permisos más complejo y la posibilidad de aplicar filtros y vistas a medida en el calendario y en los diagramas de Gantt para facilitar la búsqueda y comparación de proyectos. Además, resultaría relevante traducir la interfaz al inglés y a otros idiomas para ampliar su alcance en el ámbito europeo, desarrollar una versión móvil con modo offline-first y sincronización automática, así como explorar la integración con sistemas ERP y CRM.

10.3. Apreciaciones personales

Este proyecto ha sido un punto de inflexión en mi formación. Trabajar en un entorno empresarial real me obligó a pulir la definición de requisitos desde el primer boceto en papel hasta la versión final en código. Descubrí que entender al cliente (mediante entrevistas y prototipos rápidos) simplifica enormemente el desarrollo y

evita rehacer lo que ya tuviera hecho.

Por otro lado, enfrentarme al stack MEAN y a librerías como DHTMLX Gantt me enseñó a valorar la solidez de una buena documentación y la comunidad que respalda una tecnología. Aprendí a priorizar funcionalidades bajo presión de tiempo y comprobé lo útil que es un sistema de control de versiones y despliegue con Git y GitHub.

Al terminar, siento que he aplicado conocimientos de la carrera y que he adquirido la confianza necesaria para abordar proyectos full-stack en el ámbito profesional. Esta experiencia me deja más preparado para futuros retos y con la satisfacción de haber construido una herramienta real y útil.

BIBLIOGRAFÍA

[1] ISO, "Diagrama de gestión de proyectos en el contexto de la gobernanza de programas y carteras (ISO 21502:2020, página 5)," 2020. Extraído de la página 5 del estándar ISO 21502:2020. Consultado el 7 de junio de 2025.

[2] Moresil S.L., "Maquinaria agrícola especializada," 2025. [En línea 5/01/2025]. Disponible en: https://www.moresil.com.

[3] Smartsheet, "Menos reuniones, mayor colaboración: cómo una herramienta de gestión de proyectos ayudó a un constructor de grúas a alcanzar nuevas alturas," *The Guardian*, January 2025. [En línea 5/01/2025]. Disponible en: https://www.inkl.com/news/fewer-meetings-greater-collaboration-how-a-project-management-tool-helped-a-cran

[4] ISO, "ISO 21502:2020, Project, programme and portfolio management — Guidance on project management," 2020. Consultado el 7 de junio de 2025. Disponible de forma pública.

[5] L.-M.-U. München, "Uwe - uml-based web engineering," s.f. [En línea 5/01/2025]. Disponible en: https://uwe.pst.ifi.lmu.de/index.html.

[6] D. H. Fried, Jason y Hansson, "Basecamp: gestión sencilla de tareas y comunicación de equipos," 2004. [En línea 5/01/2025]. Disponible en: https://basecamp.com.

[7] Microsoft, "Microsoft project web access," 2007. [En línea 5/01/2025]. Disponible en: https://www.microsoft.com/project.

[8] A. Inc., "Asana: plataforma de gestión de trabajo en equipo," 2011. [En línea 5/01/2025]. Disponible en: https://asana.com.

[9] Atlassian, "Jira portfolio / advanced roadmaps," 2015. [En línea 5/01/2025]. Disponible en: https://www.atlassian.com/software/jira/portfolio.

[10] Smartsheet, "Potente administración de proyectos que prioriza a las personas," s.f. [En línea 5/01/2025]. Disponible en: https://es.smartsheet.com/solutions/project-management.

[11] Z. Corporation, "Software de gestión de proyectos en línea - zoho projects," s.f. [En línea 5/01/2025]. Disponible en: https://www.zoho.com/es-xl/projects/.

[12] Z. Corporation, "Zoho projects: una alternativa integral a smartsheet," s.f. [En línea 5/01/2025]. Disponible en: https://www.zoho.com/es-xl/projects/smartsheet-alternative.html.

[13] Google and A. Team, "What is angular?." https://angular.dev/overview, 2025. [En línea 25/07/2025].

[14] Microsoft, "Características generales y documentación de typescript," s.f. [En línea 19/05/2025]. Disponible en: https://www.typescriptlang.org.

[15] OpenJS Foundation, “About node.js®.” [En línea 25/07/2025]. Disponible en: https://nodejs.org/en/about, s.f. Descripción del entorno de ejecución asíncrono orientado a eventos y escalable, diseñado para crear aplicaciones de red.

[16] E. Project, “Express: Fast, unopinionated, minimalist web framework for node.js,” 2025. [En línea 25/07/2025]. Disponible en: https://expressjs.com/.

[17] I. MongoDB, “Mongodb database manual.” [En línea 25/07/2025]. Disponible en: https://www.mongodb.com/docs/manual/, 2025.

[18] L.-M.-U. München, “Desarrollo, estereotipos y tipado en los diagramas de la metodología uwe,” s.f. [En línea 07/05/2025]. Disponible en: https://uwe.pst.ifi.lmu.de/teachingTutorial.html.

[19] L.-M.-U. München, “Especificación del modelo de contenido siguiendo las directrices de uwe,” s.f. [En línea 19/05/2025]. Disponible en: https://uwe.pst.ifi.lmu.de/teachingTutorialContentSpanish.html.

[20] L.-M.-U. München, “Especificación del modelo de navegación siguiendo las directrices de uwe,” s.f. [En línea 19/05/2025]. Disponible en: https://uwe.pst.ifi.lmu.de/teachingTutorialNavigationSpanish.html.

[21] L.-M.-U. München, “Especificación del modelo de presentación siguiendo las directrices de uwe,” s.f. [En línea 19/05/2025]. Disponible en: https://uwe.pst.ifi.lmu.de/teachingTutorialPresentationSpanish.html.

[22] L.-M.-U. München, “Especificación del modelo de proceso siguiendo las directrices de uwe,” s.f. [En línea 19/05/2025]. Disponible en: https://uwe.pst.ifi.lmu.de/teachingTutorialProcessSpanish.html.

[23] L.-M.-U. München, “Especificación del modelado de requisitos siguiendo las directrices de uwe,” s.f. [En línea 19/05/2025]. Disponible en: https://uwe.pst.ifi.lmu.de/teachingTutorialRequirementsSpanish.html.

[24] M. L. (mattlewis92), “angular-calendar readme.” GitHub oficial, 2025. [En línea 25/07/2025]. Disponible en: https://github.com/mattlewis92/angular-calendar.

[25] DHTMLX, “Features of dhtmlx gantt library.” [En línea 5/01/2025]. Disponible en: https://dhtmlx.com/docs/products/dhtmlxGantt/features.shtml.