

UNIR FP

Informática y Comunicaciones

Técnico Superior en DAM

Título del Trabajo Fin de Estudios

Trabajo fin de estudio presentado por:	García Marcos, Miguel Ángel
Tipo de trabajo:	Gestor de Proyectos
Tutor/a:	Soy Sualdea, Damián
Fecha:	17/03/2024

Resumen

(Máximo 150 - 200 palabras)

La aplicación desarrollada tiene como objetivo la finalidad de centralizar y facilitar la gestión y supervisión de proyectos dentro de una organización. Permite a los usuarios supervisar en tiempo real el estado de cada proyecto, identificando en que fase se encuentra según las fechas establecidas. Además, proporciona una vista clara de los recursos destinados a cada proyecto, permitiendo una distribución eficiente del equipo de trabajo. Entre sus funcionalidades se incluyen la creación, edición o borrado de proyectos, fases y usuarios, así como el cálculo automático de los costes detallados y totales de cada proyecto, por separado y en conjunto. Ofrece herramientas de filtrado y búsqueda para mejorar el análisis de datos. En resumen, se trata de una plataforma adaptada a las necesidades de gestión de cualquier organización.

Palabras clave: (Máximo 5 palabras)

- Gestión de proyectos
- Usuarios
- Fases
- Costes
- Organización

Abstract

The developed application aims to centralize and facilitate the management and supervision of projects within an organization. It allows users to monitor the status of each project in real-time, identifying at what stage it is according to the established dates. Furthermore, it provides a clear view of the resources allocated to each project, enabling efficient distribution of the work team. Its functionalities include creating, editing, or deleting projects, phases, and users, as well as the automatic calculation of detailed and total costs for each project, separately and collectively. It offers filtering and search tools to enhance data analysis. In summary, it is a platform tailored to the management needs of any organization.

Keywords: (5 words max)

- Project management
- Users
- Phases
- Costs
- Organization

Índice de contenidos

1. Introducción.....	9
1.1. Justificación.....	9
1.2. Objetivos	9
2. Módulos formativos aplicados en el trabajo.....	10
3. Herramientas y lenguajes utilizados.....	11
4. Metodologías utilizadas	13
5. Componentes del equipo y aportaciones realizadas por cada alumno.....	14
5.1. Estudio de mercado	14
5.1.1. Tabla comparativa de aplicaciones actualmente en el mercado	14
5.1.2. Análisis DAFO de nuestra solución	16
5.1.3. Subapartado 1.2.....	17
5.2. Modelo de datos	17
5.3. Diagramas UML.....	20
5.3.1. Diagrama de clases	20
5.3.2. Clasificación de usuarios.....	21
5.3.3. Caso de uso 01.....	22
5.4. Diseño de interfaces.....	24
5.4.1. Wireframes	24
5.4.2. Prototipo de interfaz de alta definición	25
5.4.3. Paleta de colores.....	26
5.4.4. Logotipo	27
5.5. Planificación temporal y trabajo en equipo.....	28
5.5.1. Presupuesto temporal de tareas.....	28
5.5.2. Organización de tareas y tiempos finales	29

5.5.3.	Trabajo en equipo	30
6.	Conclusiones	31
6.1.	Análisis de desviaciones temporales y de tareas	31
6.2.	Conclusiones generales del proyecto	31
6.2.1.	Evaluación global del proyecto.	31
6.2.2.	Reflexión sobre el proceso de aprendizaje y desarrollo.	31
6.2.3.	Recomendaciones para futuros proyectos similares.....	31
6.3.	Limitaciones y prospectiva.....	32
6.3.1.	Posibles mejoras y ampliaciones del proyecto.	32
6.3.2.	Nuevas líneas de investigación o desarrollo que podrían derivarse del proyecto.32	
6.3.3.	Sugerencias para la implementación en entornos reales.....	32
7.	Referencias bibliográficas	33
Anexo A.	Diagramas de GANTT	34
Anexo B.	Código fuente de la solución y pruebas	35
Anexo C.	Manual de instalación - despliegue	36
Anexo D.	Documentación de la API	40
Anexo E.	Otros anexos de interés.....	¡Error! Marcador no definido.

Índice de figuras

Tabla 1: Herramientas, lenguajes, frameworks y APIs utilizadas	11
Tabla 2 Presupuesto temporal de tareas	¡Error! Marcador no definido.
Tabla 3 Comparativa de aplicaciones actualmente en el mercado.....	15
Ilustración 1 Análisis DAFO	16
Ilustración 2 Diagrama E/R.....	¡Error! Marcador no definido.
Ilustración 3 Diagrama de clases.....	21
Ilustración 4 Caso de uso "Recepción de pedido"	¡Error! Marcador no definido.
Ilustración 5 Wireframes.....	24
Ilustración 6 Prototipo de interfaz de alta definición	¡Error! Marcador no definido.
Ilustración 7 Paleta de colores	26
Ilustración 8 Logotipo en positivo	27
Ilustración 9 Logotipo en negativo	27

Índice de tablas

Tabla 1: Herramientas, lenguajes, frameworks y APIs utilizadas	11
Tabla 2 Presupuesto temporal de tareas	¡Error! Marcador no definido.
Tabla 3 Comparativa de aplicaciones actualmente en el mercado.....	15

1. Introducción

1.1. Justificación

Es un gestor de proyectos que sirve para mejorar, ayudar y controlar mejor cada uno de los proyectos a nivel interno de una compañía.

Con esto podremos mejorar en lo siguiente:

- Mejora en los tiempos de gestión del proyecto
- Mayor control sobre los proyectos
- Al disminuir el tiempo en la gestión de proyectos, aumenta el tiempo de la persona que controle los proyectos, para dedicarlo a otras causas de la compañía.
- Mayor control en la pérdida y seguridad de los datos.
- Mayor control sobre los empleados y en que dedican su tiempo.

1.2. Objetivos

Como comentaba en el apartado anterior al final los objetivos serán los siguientes:

- Mejora en los tiempos de gestión del proyecto
- Mayor control sobre los proyectos
- Al disminuir el tiempo en la gestión de proyectos, aumenta el tiempo de la persona que controle los proyectos, para dedicarlo a otras causas de la compañía.
- Mayor control en la pérdida y seguridad de los datos.
- Mayor control sobre los empleados y en que dedican su tiempo.

2. Módulos formativos aplicados en el trabajo

Identifica los módulos formativos y resultados de aprendizaje aplicados en el presente trabajo (consultar legislación – puedes pedir ayuda a tu tutor)

Programación

- Desarrollo de lógica de aplicación (CRUD, validaciones, control de errores).
- Uso de estructuras de datos y algoritmos básicos.

Bases de Datos

- Diseño de bases de datos relacionales (tablas, claves foráneas).
- Consulta y manipulación de datos con SQL (JOIN, INSERT, UPDATE...).

Entornos de desarrollo

- Uso de entornos como Visual Studio Code y control de versiones (Git).
- Pruebas y depuración del software.

Desarrollo de Interfaces

- Implementación de interfaces gráficas web usando React.
- Diseño adaptado a la experiencia de usuario (UX/UI).

Acceso a Datos

- Interacción entre la lógica de negocio y la base de datos a través del backend (API REST con Node.js).


Sistemas de gestión empresarial (opcional)

- Relación con la gestión de tareas y equipos en entornos reales.

3. Herramientas y lenguajes utilizados

Especifica los lenguajes, frameworks, APIs y herramientas utilizadas, junto con una breve descripción de estas.

Tabla 1: Herramientas, lenguajes, frameworks y APIs utilizadas

<p><u>Git</u></p> 	<p>Git es un sistema de control de versiones distribuido, creado por Linus Torvalds en 2005. Su propósito principal es gestionar el desarrollo del kernel de Linux, pero su flexibilidad y eficiencia lo han convertido en una herramienta ampliamente adoptada en la industria del software. Git permite a los desarrolladores rastrear cambios en el código fuente, colaborar en proyectos y revertir a versiones anteriores si es necesario. Utiliza un modelo de datos basado en instantáneas, lo que garantiza la integridad y consistencia de los datos. Además, Git facilita la creación de ramas y fusiones, lo que permite a los equipos trabajar en paralelo sin conflictos (Kranio, 2023).</p>
<p><u>IntelliJ Idea</u></p>	<p>IntelliJ IDEA es un entorno de desarrollo integrado (IDE) para Java, desarrollado por JetBrains. Ofrece soporte para el desarrollo en Spring Boot, ayudando en la programación backend gracias a su sistema de autocompletado, depuración avanzada y herramientas de gestión de proyectos.</p>
<p>Visual Studio Code (VSCode)</p>	<p>VSCode es un editor de código fuente ligero desarrollado por Microsoft. Fue utilizado para el desarrollo de la parte Frontend en React.js, permitiendo una rápida edición, instalación de extensiones y control de versiones.</p>

React.js	React es una biblioteca de JavaScript de código abierto para construir interfaces de usuario. Desarrollada por Facebook, facilita la creación de componentes reutilizables y la gestión eficiente del DOM virtual.
Spring Boot	Spring Boot es un framework de Java que simplifica la creación de aplicaciones backend. Facilita la configuración automática, el manejo de dependencias y la creación de APIs REST de forma rápida y robusta.
MySQL	MySQL es un sistema de gestión de bases de datos relacional popular en entornos de desarrollo y producción. Se utilizó para almacenar toda la información de los proyectos y fases de manera estructurada.
DBeaver	DBeaver es una herramienta de administración de bases de datos que permite gestionar, consultar y visualizar bases de datos como MySQL y PostgreSQL de forma gráfica. Fue utilizado para la gestión de tablas, consultas SQL y depuración de datos.
npm (Node Package Manager)	npm es el sistema de gestión de paquetes de Node.js. Se utilizó para instalar las dependencias necesarias para el proyecto Frontend, como React Icons, React Router, Bootstrap, etc.

Bootstrap	Bootstrap es un framework de CSS que permite el diseño de aplicaciones web responsivas y estilizadas de forma rápida. Se utilizó para estructurar el Frontend de manera ordenada.
Spring Data JPA	Es una parte de Spring que facilita el acceso a bases de datos mediante la abstracción de repositorios. Permite realizar operaciones CRUD sin necesidad de implementar consultas SQL manuales.
Postman	Postman es una herramienta que permite realizar pruebas de APIs REST. Se usó para probar manualmente los endpoints creados en el backend, enviando solicitudes GET, POST, PUT y DELETE.

4. Metodologías utilizadas

Dado que este proyecto ha sido realizado de forma individual y en paralelo con responsabilidades laborales, no se ha seguido una metodología de desarrollo en específico como Scrum o Kanban. Sin embargo, se ha trabajado de manera iterativa y adaptativa, resolviendo cada parte del sistema por fases según el tiempo disponible y la complejidad de cada componente.

El enfoque fue modular, desarrollando primero la base de datos, luego una base de la API backend y finalmente combinando el Front con API backend. Se priorizó tener funcionalidad completas y probadas antes de avanzar.

Aunque no se emplearon herramientas de gestión de proyectos, se utilizaron listas de tareas personales y controles de versiones con Git para mantener el orden y coherencia del desarrollo.

Este enfoque mas flexible ha permitido compatibilizar el trabajo profesional con el avance del TFC, logrando una solución funcional y estructurada dentro de estas limitaciones.

5. Componentes del equipo y aportaciones realizadas por cada alumno

Este proyecto ha sido desarrollado íntegramente de forma individual. Esto ha implicado asumir todos los roles del proyecto como análisis, diseño, desarrollo y pruebas.

Rol asumido	Tareas realizadas
Analista	Estudio de necesidades, definición de funcionalidades, modelado de datos
Diseñador	Diseño de interfaces, elección de paleta de colores y estructura visual
Desarrollador Backend	Programación de la API REST, conexión con la base de datos, lógica de negocio
Desarrollador Frontend	Implementación en React, interactividad, gestión de estados y validaciones
Tester	Pruebas funcionales, revisión de errores, mejoras iterativas

5.1. Estudio de mercado

Estudio de la temática a desarrollar, análisis de la competencia y mejoras a realizar, estudio del usuario de la aplicación.

5.1.1. Tabla comparativa de aplicaciones actualmente en el mercado

Tabla 2 Comparativa de aplicaciones actualmente en el mercado

Característica	Trello	Asana	Taskmanager
Descripción	App de gestión visual de tareas	App de planificación de proyectos	Web app para gestionar proyectos
Funcionalidades	Tableros, listas, tarjetas	Tareas, cronogramas, dependencias	CRUD proyectos, fases y usuarios
Público Objetivo	Equipos pequeños / medianos	Empresas de tamaño medio	PYMEs, proyectos académicos
Plataformas	Web, iOS, Android	Web, iOS, Android	Web
Precio	Freemium	Gratuito limitado / premium	Gratuita
Puntuación	4.5 / 5.0	4.6 / 5.0	4.7 / 5.0 (estimada)

5.1.2. Análisis DAFO de nuestra solución



Ilustración 1 Análisis DAFO

Debilidades

- Requiere conexión a Internet para su funcionamiento.
- No cuenta con una aplicación móvil nativa que permita acceder desde smartphones.
- La interfaz es funcional pero menos pulida que otras soluciones comerciales más maduras.

Amenazas

- Alta competencia en el mercado con herramientas consolidadas como Trello, Asana o Microsoft Project.
- Posibles vulnerabilidades de seguridad si la aplicación no se actualiza regularmente.
- Evolución constante de las tecnologías y frameworks, lo que exige mantenimiento continuo.

Fortalezas

- Interfaz sencilla e intuitiva, fácilmente personalizable según las necesidades del usuario.
- Ofrece control detallado sobre usuarios, fases del proyecto y costes asociados.
- Adecuada para pequeñas empresas, centros educativos o equipos reducidos dentro de organizaciones.

Oportunidades

- Potencial para desarrollar una versión móvil multiplataforma.
- Posibilidad de implementar gráficos interactivos para visualización de datos.
- Capacidad de evolucionar hacia un producto comercial escalable o modelo SaaS.

5.1.3. Subapartado 1.2

Texto Normal del menú de estilos.

5.2. Modelo de datos

Diagrama E/R:

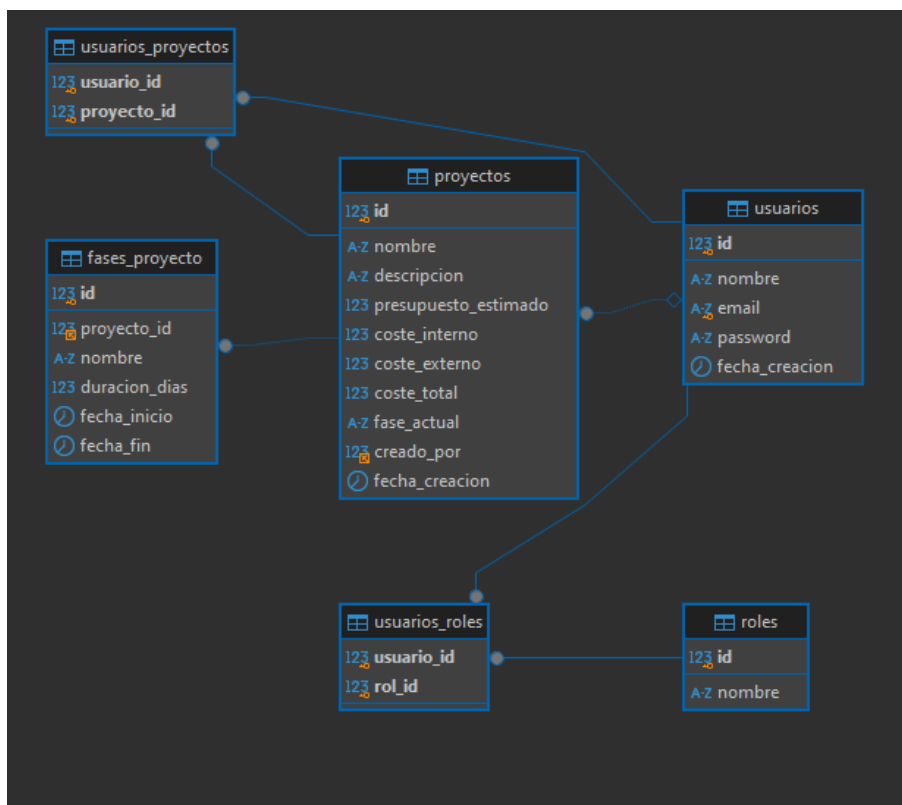


Tabla Usuarios:

Contiene la información de cada usuario del sistema.

Campo	Tipo	Descripción
<code>id</code>	INT, PK, AI	Identificador único del usuario.
<code>nombre</code>	VARCHAR(100)	Nombre completo del usuario.
<code>email</code>	VARCHAR(100), UNIQUE	Correo electrónico del usuario.
<code>password</code>	VARCHAR(255)	Contraseña cifrada del usuario.
<code>fecha_creacion</code>	TIMESTAMP	Fecha de creación del usuario.

Tabla proyectos:

Representa cada proyecto registrado en el sistema.

Campo	Tipo	Descripción
<code>id</code>	INT, PK, AI	Identificador del proyecto.
<code>nombre</code>	VARCHAR(100)	Nombre del proyecto.
<code>descripcion</code>	VARCHAR(300)	Descripción detallada.
<code>presupuesto_estimado</code>	DECIMAL(10,2)	Presupuesto asignado.
<code>coste_interno</code>	DECIMAL(10,2)	Coste interno estimado.
<code>coste_externo</code>	DECIMAL(10,2)	Coste externo estimado.
<code>coste_total</code>	DECIMAL(10,2)	Suma de los costes internos y externos.
<code>fase_actual</code>	VARCHAR(100)	Fase actual del proyecto (calculada por fechas).
<code>creado_por</code>	INT	ID del usuario que creó el proyecto. FK a <code>usuarios(id)</code> .
<code>fecha_creacion</code>	TIMESTAMP	Fecha de creación del proyecto.

Tabla usuarios_proyectos:

Asigna uno o varios usuarios a cada proyecto.

Campo	Tipo	Descripción
usuario_id	INT	ID del usuario. FK a usuarios(id) .
proyecto_id	INT	ID del proyecto. FK a proyectos(id) .

Tabla fases_proyecto:

Define las fases que componen un proyecto, con sus fechas y la duración de cada una de ellas.

Campo	Tipo	Descripción
id	INT, PK, AI	Identificador de la fase.
proyecto_id	INT	Proyecto al que pertenece. FK a proyectos(id) .
nombre	VARCHAR(100)	Nombre de la fase (Requisitos , Desarrollo , etc.).
duracion_dias	INT	Duración estimada.
fecha_inicio	DATE	Fecha de inicio real.
fecha_fin	DATE	Fecha de fin real.

Tabla usuarios_roles:

Tabla intermedia para gestionar la relación entre usuarios y roles. Ya que un usuario puede tener varios roles y varios roles pueden ser gestionados por un usuario.

Campo	Tipo	Descripción
usuario_id	INT (FK)	Identificador del usuario.
rol_id	INT (FK)	Identificador del rol.

Tabla roles:

Esta tabla almacena los diferentes tipos de roles que puede tener un usuario del sistema.

Campo	Tipo	Descripción
id	INT (PK)	Identificador único del rol.
nombre	VARCHAR(100)	Nombre del rol (por ejemplo, "Administrador", "Desarrollador").

5.3. Diagramas UML

Los diagramas UML (Unified Modeling Language) son una herramienta visual que permite representar el diseño y estructura de un sistema informático. Son esenciales para documentar cómo interactúan las distintas partes del software, tanto desde el punto de vista estructural (diagrama de clases) como funcional (casos de uso).

En este proyecto, se han aportado los siguientes diagramas:

- Diagrama de clases
- Diagrama de casos de uso

5.3.1. Diagrama de clases

El diagrama de clases representa la estructura de las entidades del sistema, sus atributos y relaciones. Las clases son:

- Usuario: contiene atributos como id, nombre, email y password.
- Rol: define los distintos tipos de roles que puede tener un usuario.
- Proyecto: agrupa la información relacionada con los proyectos, como su nombre, costes, fase_actual y su relación con el usuario creador del proyecto.
- Fases Proyecto: representa cada etapa del proyecto, con fechas de inicio y fin de cada una de las fases, y su duración.

Relaciones:

- Usuario y Proyecto: Muchos a muchos (a través de tabla intermedia).
- Usuario y Rol: Muchos a muchos
- Proyecto y Fase Proyecto: Uno a muchos.

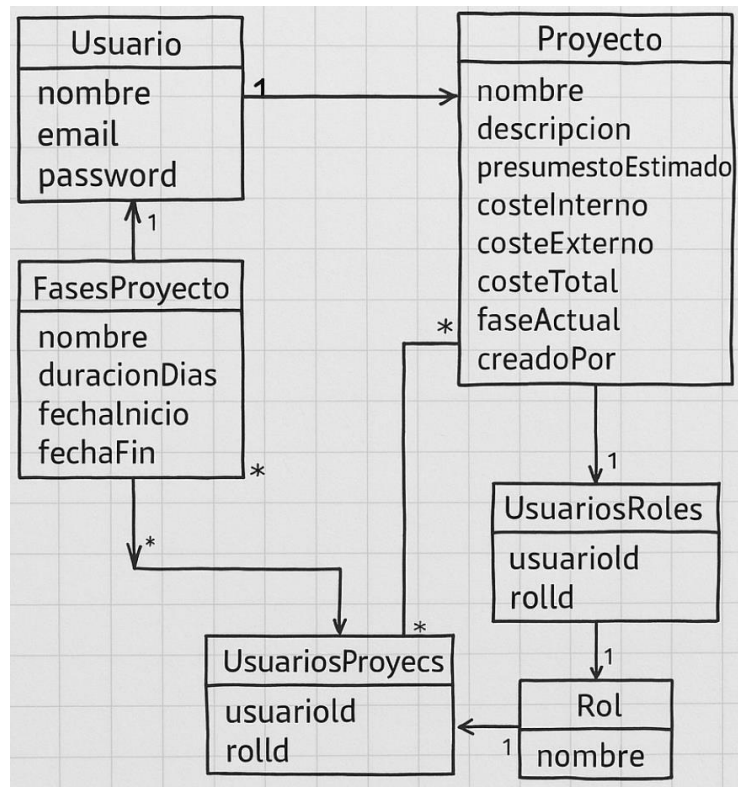


Ilustración 2 Diagrama de clases

5.3.2. Clasificación de usuarios

En nuestra aplicación se contemplan distintos tipos de usuarios, cada uno definido con un rol.

Para comprender mejor estos perfiles, se ha aplicado la metodología Persona del enfoque Design Thinking, la cual permite representar a los usuarios mediante estereotipos realistas basados en sus comportamientos.

Jefe de Proyecto:

- **Objetivo:** Tener una visión clara del avance de cada proyecto.
- **Funciones clave:** Crear proyectos, asignar fases y usuarios, controlar costes.
- **Necesidades:** Informes rápidos, interfaz clara, control del equipo.

Desarrollador:

- **Objetivo:** Saber en que proyectos está trabajando y en qué fases.
- **Funciones clave:** Consultar proyectos asignados y tareas en función de las fases.

- **Necesidades:** Acceso sencillo y rápido, sin mucha información.

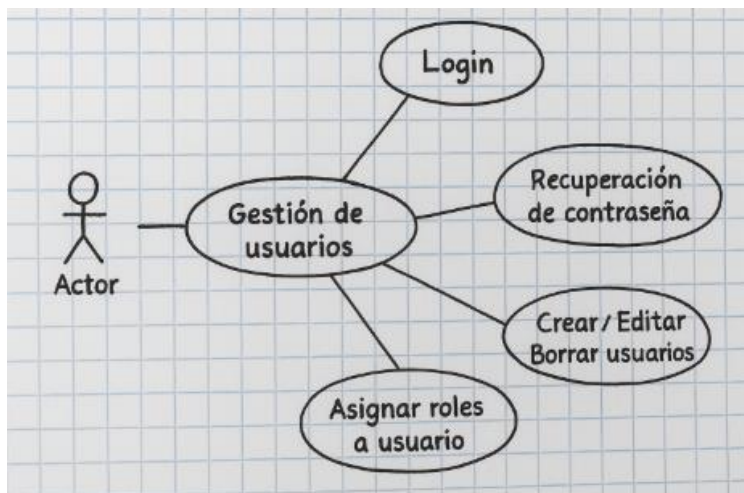
Administrador del sistema:

- **Objetivo:** Gestionar la aplicación, nuevos usuarios, roles y mantener la seguridad.
- **Funciones clave:** Control de accesos y creación de usuarios, roles.
- **Necesidades:** Seguridad y trazabilidad del sistema.

5.3.3. Caso de uso 01

Gestión de usuarios

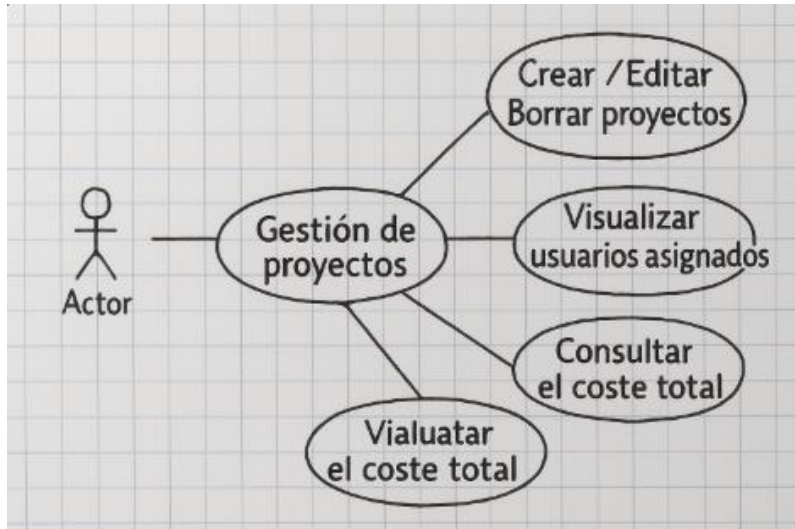
- **Login:** autenticación mediante correo y contraseña.
- **Recuperación de contraseña:** si un usuario olvida la contraseña, el sistema genera una nueva automáticamente y la envía por correo. Esta contraseña se actualiza directamente en la base de datos.
- **Crear / Editar / Borrar usuarios.**
- **Asignar roles** a cada usuario (por ejemplo: Jefe de Proyecto, Desarrollador, QA, etc.).



Gestión de proyectos

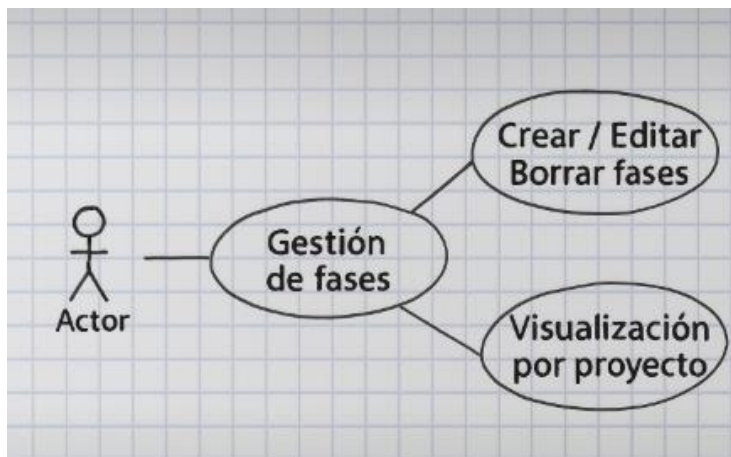
- **Crear / Editar / Borrar proyectos.**
- **Asignar usuarios** a un proyecto.
- **Visualizar usuarios asignados.**

- **Consultar el coste total** (calculado automáticamente como suma de coste interno y externo).



Gestión de fases

- **Crear / Editar / Borrar fases** para cada proyecto.
- **Control automático de la fase actual** según la fecha actual.
- **Visualización por proyecto** de todas las fases asociadas.



Filtrado y consulta

- Filtrar proyectos por:
 - Nombre, presupuesto, costes, fase actual, etc.
- Filtrar usuarios por:
 - Nombre, correo electrónico, roles, etc.

- Consulta agregada de **costes totales** de todos los proyectos.



Sesiones

- **Logout:** cierre seguro de sesión.

5.4. Diseño de interfaces

5.4.1. Wireframes

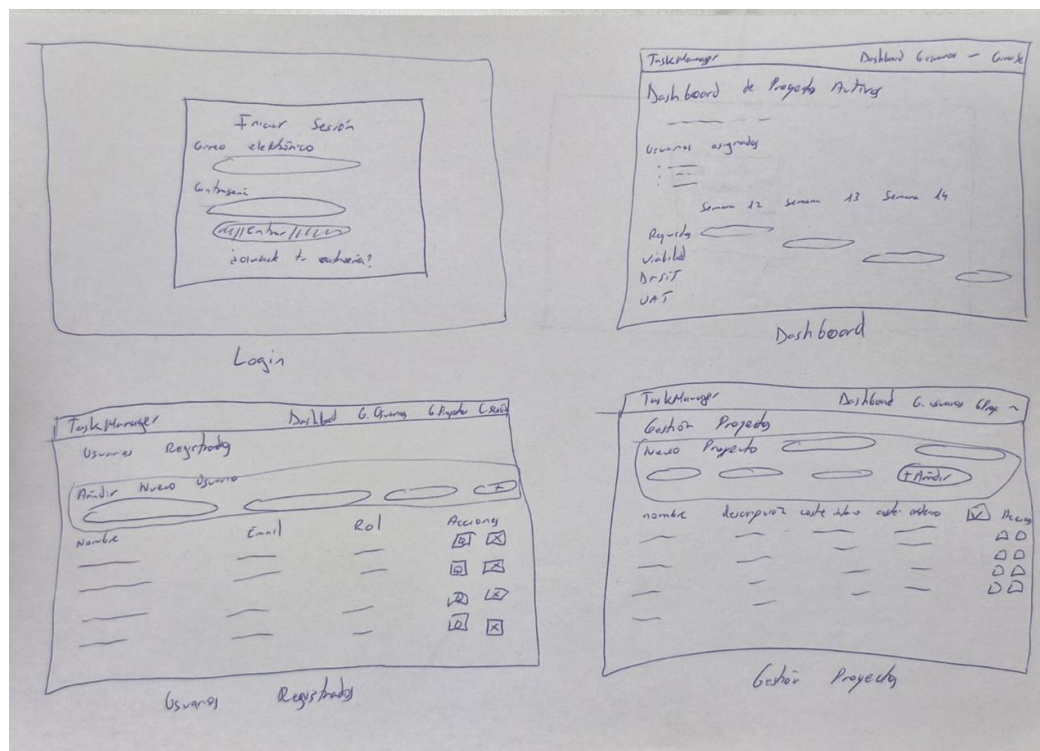
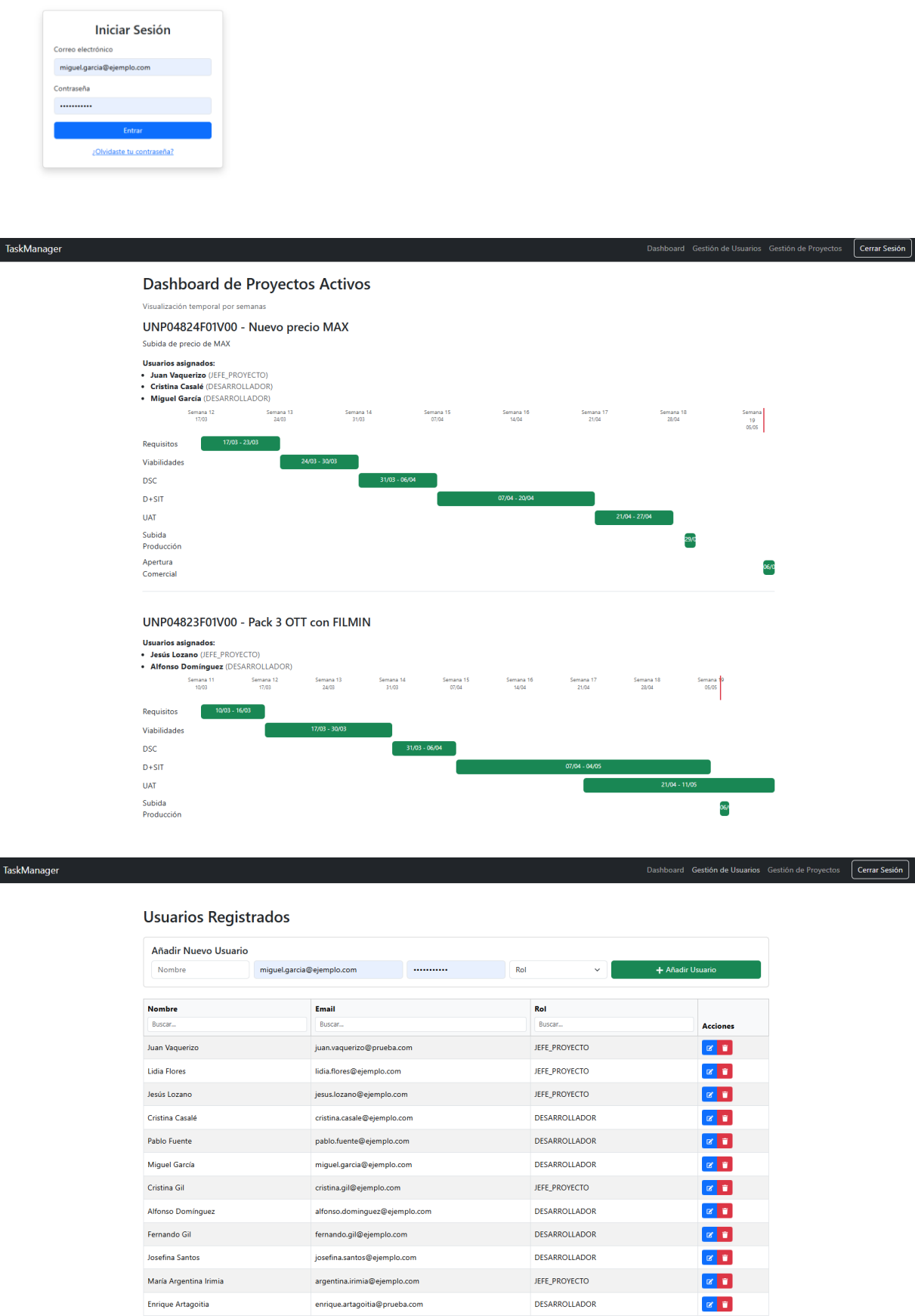


Ilustración 3 Wireframes

5.4.2. Prototipo de interfaz de alta definición



TaskManager

DashboardGestión de UsuariosGestión de ProyectosCerrar Sesión

Gestión de Proyectos

Nuevo Proyecto

nombre

descripcion

presupuesto estimado

coste interno

coste externo

coste total

Selecciona fase

+ Añadir Proyecto

nombre	descripcion	presupuesto estimado	coste interno	coste externo	coste total	fase actual	creadoPor		
Filtrar	Filtrar	Filtrar	Filtrar	Filtrar	Filtrar	Filtrar	Filtrar	Fases	Acciones
UNP04824F01V00 - Nuevo precio MAX	Subida de precio de MAX	25424.68	10540	7910	18450	Sin fase	Juan Vaquerizo		
Fases del Proyecto									
Nombre	Duración	Inicio	Fin	Acciones					
Requisitos	7	2025-03-17	2025-03-23						
Viabilidades	7	2025-03-24	2025-03-30						
DSC	7	2025-03-31	2025-04-06						
D+SIT	14	2025-04-07	2025-04-20						
UAT	7	2025-04-21	2025-04-27						
Subida Producción	1	2025-04-29	2025-04-29						
Apertura Comercial	1	2025-05-06	2025-05-06						
Selecciona fase	1	dd/mm/aaaa	dd/mm/aaaa						
Usuarios Asignados									
Nombre	Email	Acciones							
Juan Vaquerizo	juan.vaquerizo@prueba.com								
Cristina Casale	cristina.casale@ejemplo.com								
Miguel García	miguel.garcia@ejemplo.com								
Selecciona usuario									

5.4.3. Paleta de colores

He utilizado esta paleta de colores, ya que, en su conjunto, ofrece equilibrio visual, usabilidad y una apariencia profesional, facilitando así la experiencia de usuario.

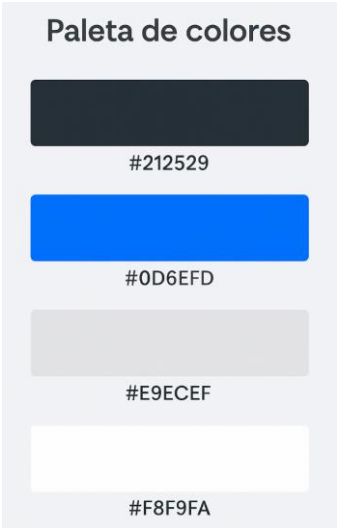


Ilustración 4 Paleta de colores

5.4.4. Logotipo



Ilustración 5 Logotipo en positivo

Logotipo en negativo



Ilustración 6 Logotipo en negativo

5.5. Planificación temporal y trabajo en equipo

5.5.1. Presupuesto temporal de tareas

Tareas	Subtareas	Duración (horas)	Persona/s asignada/s
Planificación del Proyecto	Definir objetivos y alcance	4h	Miguel
	Estudio de mercado y análisis de competencia	3h	Miguel
	Análisis DAFO y casos de uso	2h	Miguel
Diseño de la Aplicación	Diseño arquitectura de BD	6h	Miguel
	Creación de diagramas UML	3h	Miguel
	Bocetos y diseño UI	4h	Miguel
Desarrollo Frontend	Implementar interfaz de usuario	18h	Miguel
	Gestión de estado y eventos	5h	Miguel
	Realizar pruebas de usabilidad	1h	Miguel
Desarrollo Backend	Configurar servidor y BD	8h	Miguel
	Implementar lógica de negocio	8h	Miguel
	Crear APIs y servicios web	8h	Miguel
Integración y Pruebas	Integrar frontend y backend	8h	Miguel
	Realizar pruebas funcionales	4h	Miguel
	Corregir errores y optimizar	4h	Miguel
Despliegue y Lanzamiento	Configurar entorno de producción	1h	Miguel
	Desplegar la aplicación	1h	Miguel

	Realizar pruebas finales	1h	Miguel
--	--------------------------	----	--------

5.5.2. Organización de tareas y tiempos finales

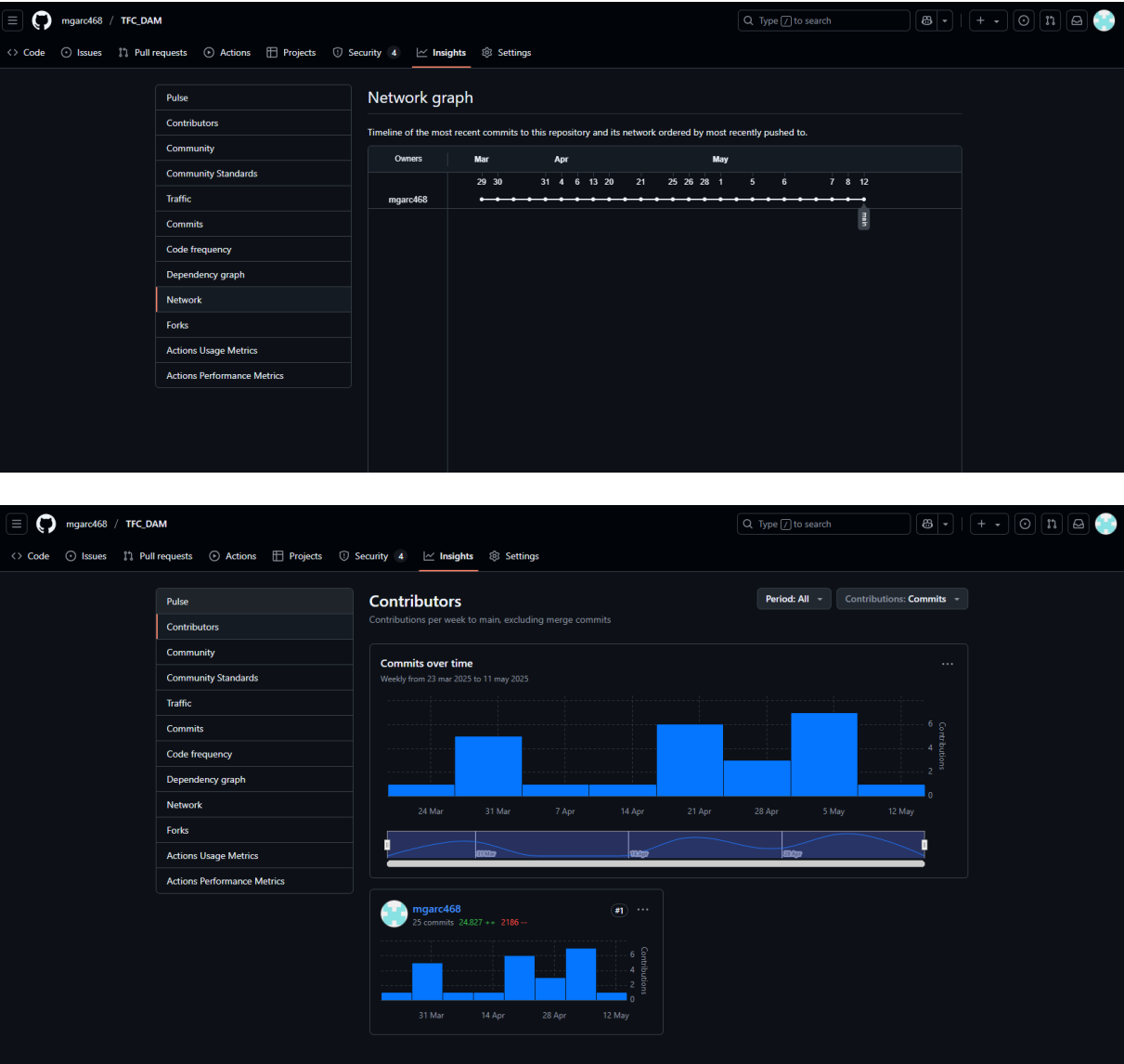
Tareas	Subtareas	Duración (horas)	Persona/s asignada/s
Planificación del Proyecto	Definir objetivos y alcance	4h	Miguel
	Estudio de mercado y análisis de competencia	2h	Miguel
	Análisis DAFO y casos de uso	2h	Miguel
Diseño de la Aplicación	Diseño arquitectura de BD	5h	Miguel
	Creación de diagramas UML	3h	Miguel
	Bocetos y diseño UI	4h	Miguel
Desarrollo Frontend	Implementar interfaz de usuario	16h	Miguel
	Gestión de estado y eventos	5h	Miguel
	Realizar pruebas de usabilidad	2h	Miguel
Desarrollo Backend	Configurar servidor y BD	8h	Miguel
	Implementar lógica de negocio	8h	Miguel
	Crear APIs y servicios web	6h	Miguel
Integración y Pruebas	Integrar frontend y backend	3h	Miguel
	Realizar pruebas funcionales	4h	Miguel
	Corregir errores y optimizar	4h	Miguel
	Configurar entorno de producción	1h	Miguel

Despliegue y Lanzamiento	Desplegar la aplicación	1h	Miguel
	Realizar pruebas finales	1h	Miguel

En los anexos incorpora el diagrama de Gantt correspondiente al presupuesto y el resultante al final del proyecto.

5.5.3. Trabajo en equipo

Gráfico de aportaciones:



6. Conclusiones

6.1. Análisis de desviaciones temporales y de tareas

Realiza un análisis de las diferencias entre el presupuesto inicial de tareas y tiempos y las tareas y tiempos realizados finalmente.

Puedes apoyarte en una tabla comparativa, comentando los motivos de las posibles desviaciones.

6.2. Conclusiones generales del proyecto

6.2.1. Evaluación global del proyecto.

La evaluación general es bastante positiva. A pesar de que el desarrollo se ha realizado de forma individual y con limitaciones de tiempo por motivos laborales, se ha conseguido completar los objetivos planteados inicialmente. Cumple con los criterios básicos de una aplicación CRUD moderna. El sistema es estable y escalable para seguir con su futuro desarrollo.

6.2.2. Reflexión sobre el proceso de aprendizaje y desarrollo.

Me ha permitido consolidar conocimientos y también aprender nuevas tecnologías como React, ya que hemos podido ver muy poco de esta parte en clase. He aprendido a diseñar una aplicación desde cero, juntando todos los conocimientos del ciclo formativo.

También he aprendido a valorar el esfuerzo que conlleva realizar un desarrollo completo de forma autónoma.

6.2.3. Recomendaciones para futuros proyectos similares.

Para futuros proyectos similares, recomendaría realizar una planificación de todo por horas de desarrollo frente a las horas disponibles. Documentar a la misma vez que se va realizando la aplicación y se va desarrollando, tanto en código como realizando la documentación oportuna del proyecto. Es muy recomendable, mantener una estructura clara desde el principio tanto a nivel de carpetas como de la lógica de código, para facilitar enormemente el mantenimiento y futuras ampliaciones.

6.3. Limitaciones y prospectiva

6.3.1. Posibles mejoras y ampliaciones del proyecto.

- Desarrollo de una aplicación móvil nativa para Android/iOS.
- Gestión de permisos según el rol de usuario.
- Exportación de informes en Excel, útiles para reportes de documentación.
- Dashboard visual con gráficas dinámicas, para representar costes y progreso por fase de manera intuitiva.
- Notificaciones por correo ante cambios en fases o asignaciones de usuario.
- Nuevo estado, sobre en que estado se encuentra el proyecto.
- Comentarios en el dashboard, con fechas para hacer seguimiento.
- Mejora de la seguridad con contraseñas cifradas.
- Añadir nuevo módulo para guardar la documentación de la realización de pruebas, tanto las de desarrollo como las de usuarios.

6.3.2. Nuevas líneas de investigación o desarrollo que podrían derivarse del proyecto.

- Análisis de rendimiento de equipos: A través de métricas de tiempo invertido por fase y usuario.
- Integración con servicios cloud (Google Drive, Dropbox) para almacenar documentación del proyecto directamente desde la aplicación.

6.3.3. Sugerencias para la implementación en entornos reales.

- Alojamiento seguro en servidores profesionales con acceso restringido y políticas de backup automático.
- Cifrado robusto de contraseñas y comunicaciones (HTTPS, JWT, Bcrypt) para garantizar la protección de datos personales.
- Documentación técnica completa para facilitar el mantenimiento y posibles integraciones futuras.
- Formación básica para los usuarios finales sobre el uso del sistema y flujos de trabajo.
- Sistema de roles bien definido con validaciones en backend para evitar accesos indebidos.

7. Referencias bibliográficas

Design Thinking Services. (2023). *Método persona*. (D. T. Services, Ed.) Recuperado el 01 de 10 de 2024, de <https://www.designthinking.services/herramientas-design-thinking/metodo-persona/>

Figma. (2023). *Figma: Herramienta de diseño de interfaces*. Recuperado el 20 de 10 de 2024, de <https://www.figma.com>

Kranio. (29 de 08 de 2023). *Descubriendo Git: Características y Ventajas*. Recuperado el 10 de 2024, de <https://www.kranio.io/blog/descubriendo-git-caracteristicas-y-ventajas>

MySQL. (2024). *MySQL Reference Manual*. Recuperado de <https://dev.mysql.com/doc/>

React.js. (2024). Documentación oficial. Recuperado de <https://reactjs.org>

Oracle. (2024). *Documentación oficial de Java SE*. Recuperado de <https://docs.oracle.com/javase/>

Microsoft. (2024). *Visual Studio Code Documentation*. Recuperado de <https://code.visualstudio.com/docs>

ANEXO A. DIAGRAMAS DE GANTT

Gráfico Temporal:

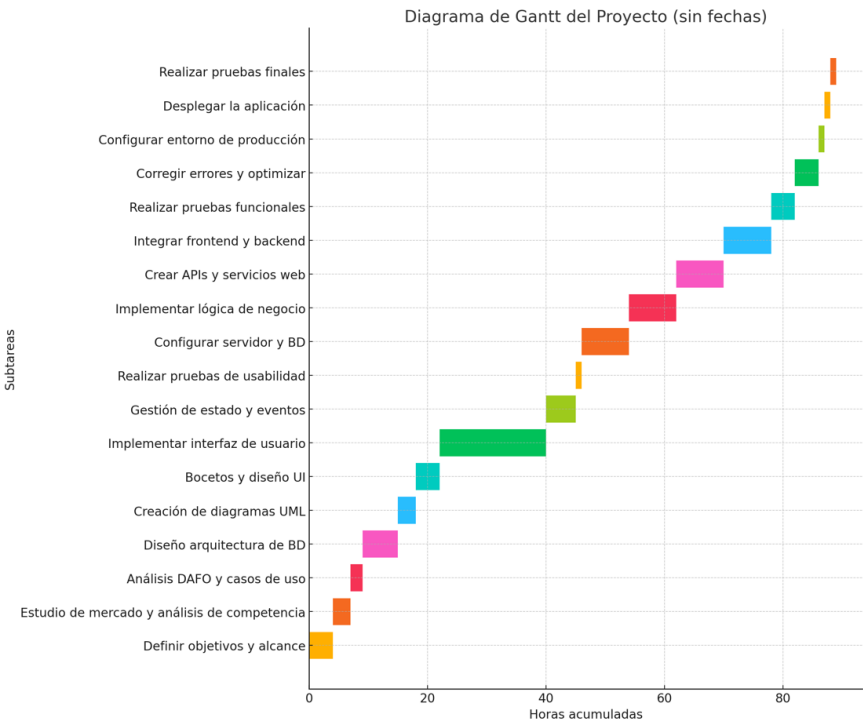
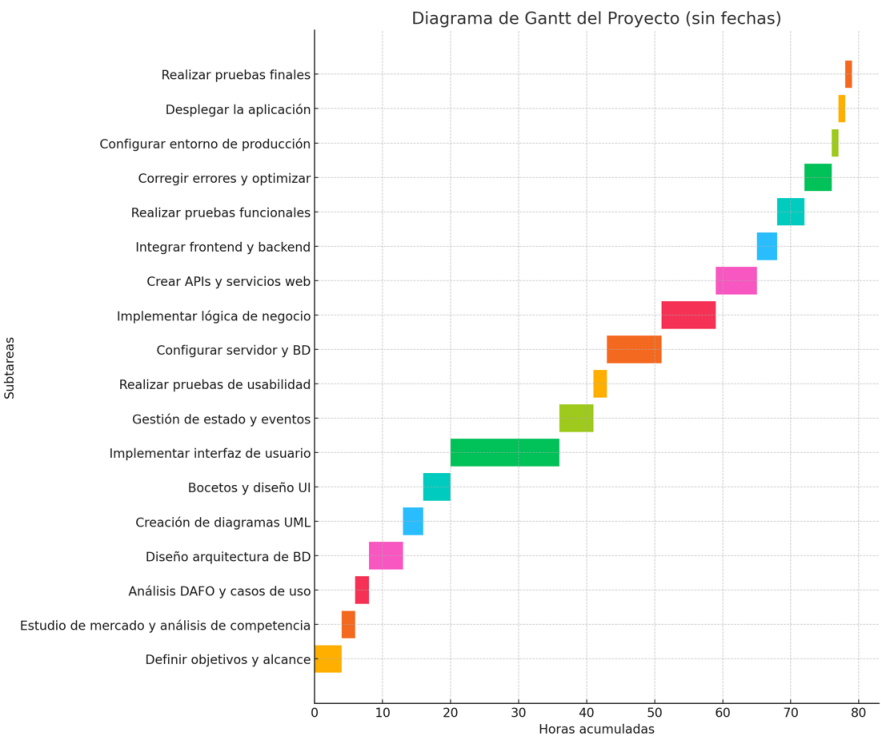


Gráfico Final:



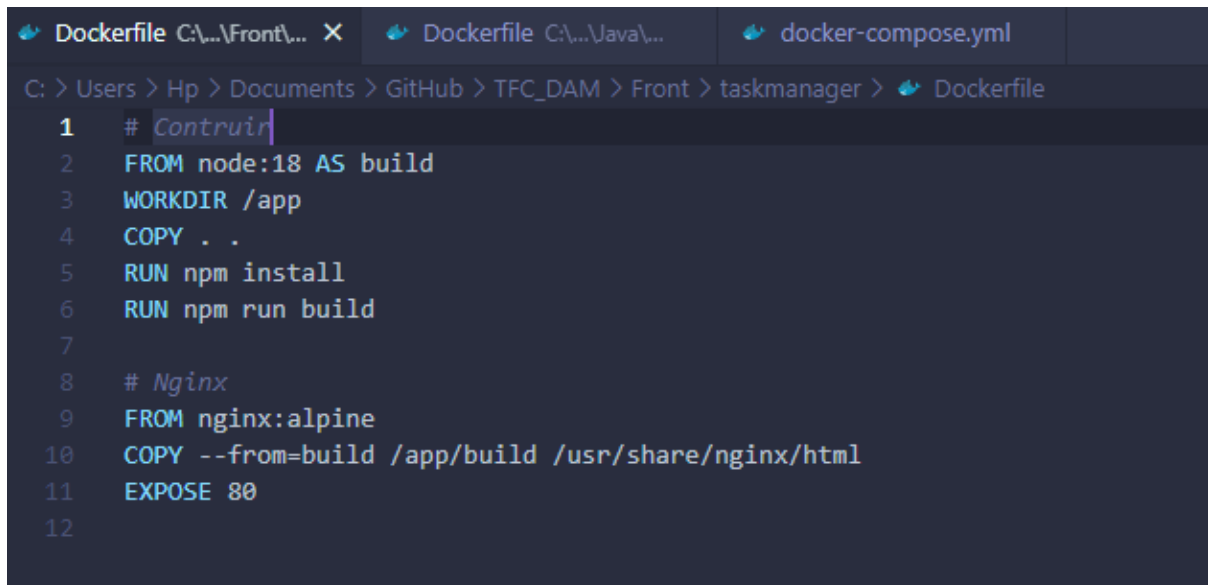
ANEXO B. CÓDIGO FUENTE DE LA SOLUCIÓN Y PRUEBAS

Pruebas realizadas desde el Front y desde Postman:

- Comprobación del login, tanto OK, como FAIL.
- Recuperación de contraseña.
- Alta usuario.
- Modificar usuario.
- Eliminar usuario.
- Alta proyecto.
- Modificar proyecto.
- Eliminar proyecto. (Probar también con uno que tenga fases para que se elimine todo, incluso si hay usuarios asignados a ese proyecto).
- Crear fase.
- Modificar fase.
- Eliminar fase.
- Comprobación del dashboard. (Ver que coinciden los proyectos, activos y sus fases).

ANEXO C. MANUAL DE INSTALACIÓN – DESPLIEGUE

1. Añadir archivo Dockerfile en el back.



The screenshot shows a code editor with three tabs: 'Dockerfile C:\...\Front\...', 'Dockerfile C:\...\Java\...', and 'docker-compose.yml'. The active tab is 'Dockerfile C:\...\Java\...'. The file path is 'C: > Users > Hp > Documents > GitHub > TFC_DAM > Front > taskmanager > Dockerfile'. The Dockerfile content is as follows:

```
1 # Contruir
2 FROM node:18 AS build
3 WORKDIR /app
4 COPY . .
5 RUN npm install
6 RUN npm run build
7
8 # Nginx
9 FROM nginx:alpine
10 COPY --from=build /app/build /usr/share/nginx/html
11 EXPOSE 80
12
```

2. Añadir archivo Dockerfile en el front.



The screenshot shows a code editor with three tabs: 'Dockerfile C:\...\Front\...', 'Dockerfile C:\...\Java\...', and 'docker-compose.yml'. The active tab is 'Dockerfile C:\...\Java\...'. The file path is 'C: > Users > Hp > Documents > GitHub > TFC_DAM > Java > taskmanager > Dockerfile'. The Dockerfile content is as follows:

```
1 FROM openjdk:17-jdk-slim
2 VOLUME /tmp
3 COPY target/taskmanager-0.0.1-SNAPSHOT.jar app.jar
4 ENTRYPOINT ["java", "-jar", "/app.jar"]
5
```

3. Construir de nuevo el proyecto taskmanager del back, para generar el .jar.

[illegible]

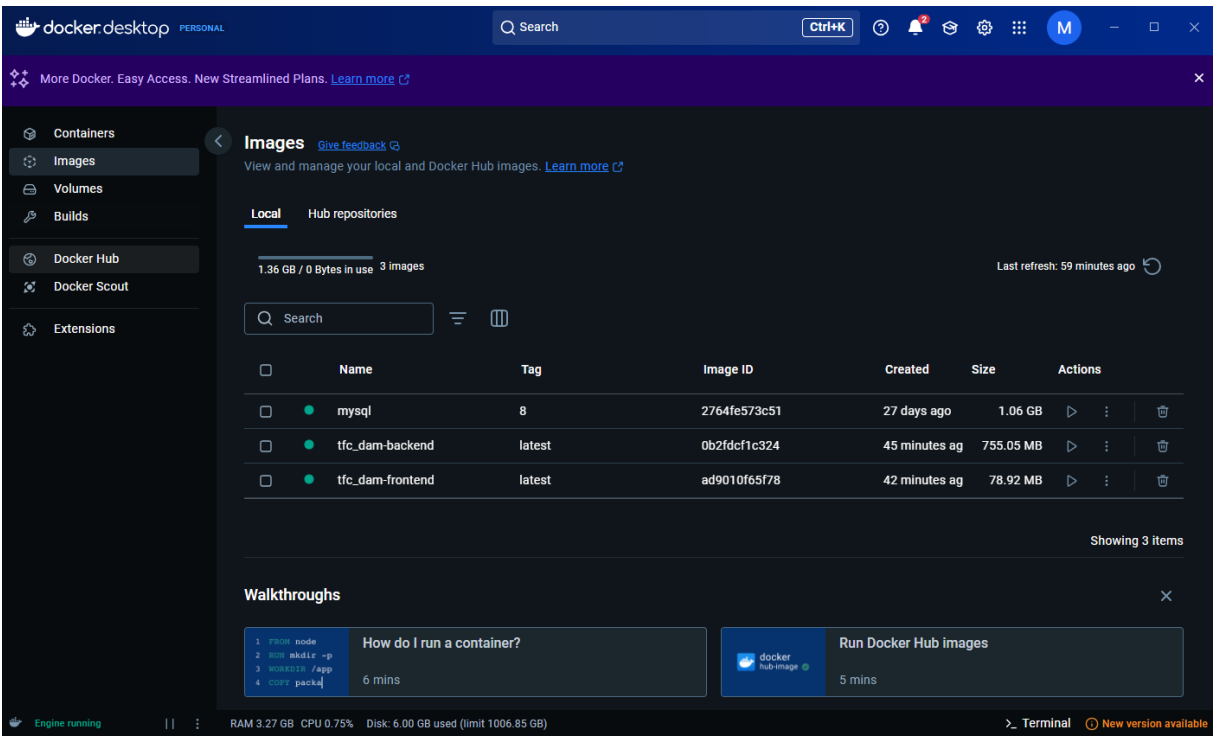
4. Crear archivo .yaml

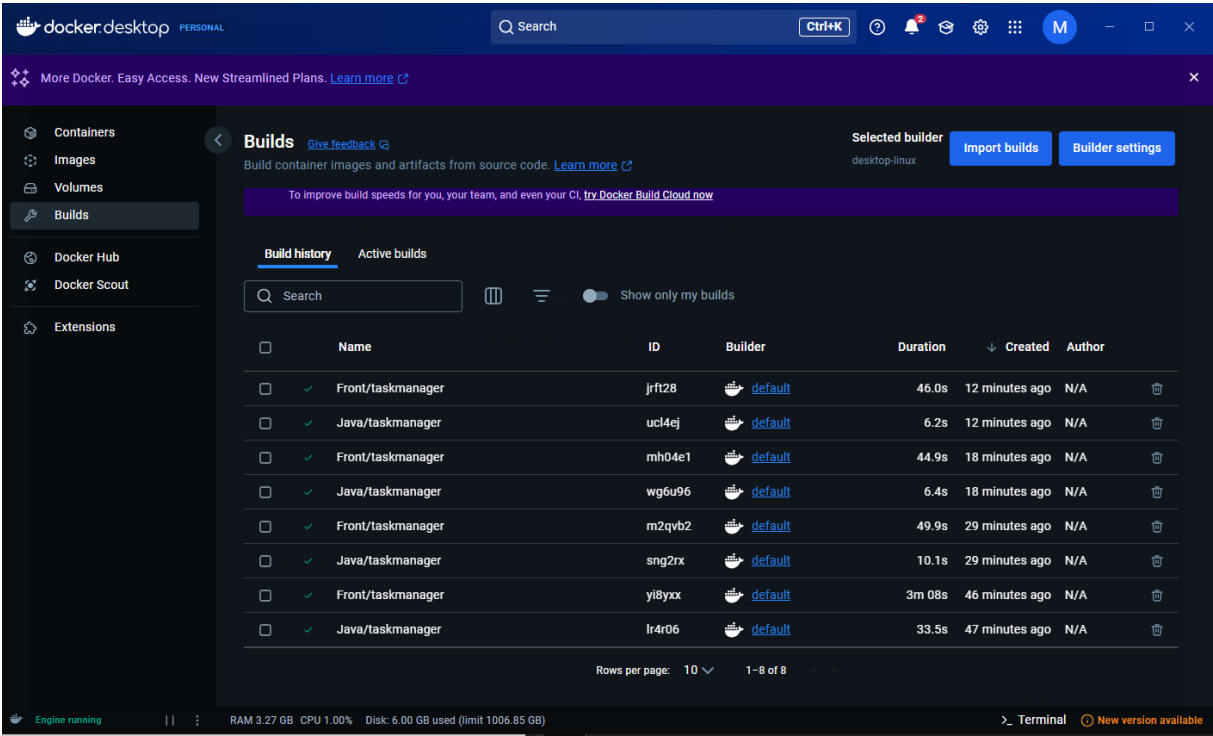
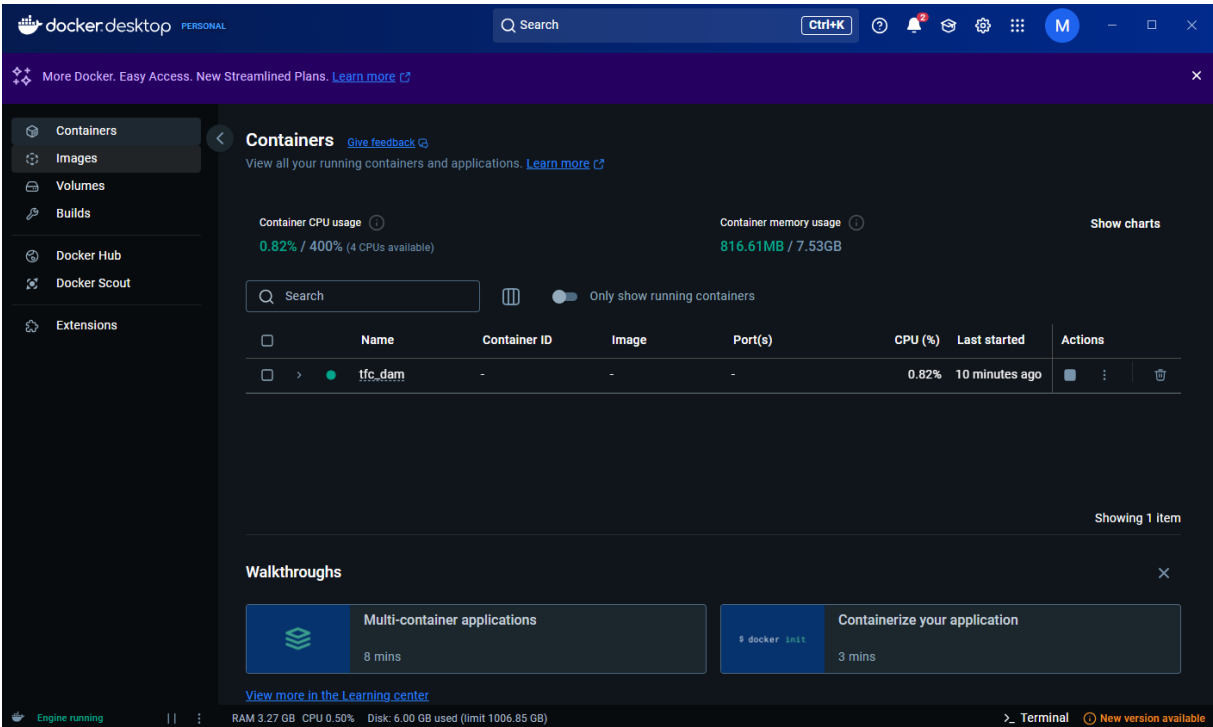
```
C:\Users > Hp > Documents > GitHub > TFC_DAM > docker-compose.yml
```

```
1  services:
2    db:
3      image: mysql:8
4      container_name: mysql_db
5      environment:
6        MYSQL_DATABASE: taskmanager
7        MYSQL_ALLOW_EMPTY_PASSWORD: "yes"
8      ports:
9        - "3306:3306"
10     volumes:
11       - db_data:/var/lib/mysql
12     healthcheck:
13       test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
14       interval: 10s
15       timeout: 5s
16       retries: 5
17
18   backend:
19     build:
20       context: ./Java/taskmanager
21     container_name: backend_app
22     environment:
23       SPRING_DATASOURCE_URL: jdbc:mysql://db:3306/taskmanager
24       SPRING_DATASOURCE_USERNAME: root
25       SPRING_DATASOURCE_PASSWORD:
26     ports:
27       - "8080:8080"
28     depends_on:
29       db:
30         condition: service_healthy
31     restart: always
32
33   frontend:
34     build:
35       context: ./Front/taskmanager
36     container_name: react_app
37     ports:
38       - "3000:80"
39     depends_on:
40       - backend
41     restart: always
42
43 volumes:
44   db_data:
```

5. Construir y levantar el proyecto completo

```
...
C:\Users\wp\Documents\gitHub\TFC_Dam-docker-compose up --build
time=2025-08-11T11:22:54+02:00 level=warning msg=C:\Users\wp\Documents\gitHub\TFC_DAM\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Building 223.4s (25/25) FINISHED
[backend internal] load build definition from Dockerfile
--> transferring dockerfile: 173B
[backend internal] load metadata for docker.io/library/openjdk:17-jdk-slim
[backend auth] library/openjdk:pull token for registry-1.docker.io
[backend internal] load .dockerignore
[backend internal] load build context
--> exporting layers
[backend 1/2] FROM docker.io/library/openjdk:17-jdk-slim@sha256:aaa3b3cb27e3e52808f1686180580c438ed5ecfa8bc126041f68c3f62f9774
--> resolve docker.io/library/openjdk:17-jdk-slim@sha256:aaa3b3cb27e3e52808f1686180580c438ed5ecfa8bc126041f68c3f62f9774
--> sha256:6c99df4f680b0d12347f0328a5d0988a70a727056 187.99MB / 187.99MB
--> sha256:1fe172e49f0830b4541a26174112bc119f9fec42a650db0b9491aee3233 31.39MB / 31.39MB
--> sha256:44d3a8b06767549d851880bcd9dae718f4e4ff4b0b042209cf384e59100 1.58MB / 1.58MB
--> extracting sha256:1fe172e49f0830b4541a26174112bc119f9fec42a650db0b9491aee3233
--> extracting sha256:44d3a8b06767549d851880bcd9dae718f4e4ff4b0b042209cf384e59100
--> extracting sha256:6c99df4f680b0d12347f0328a5d0988a70a727056
[backend 2/2] COPY target/taskmanager-0.4.1-SNAPSHOT.jar app.jar
[backend] exporting to image
--> exporting layers
--> exporting manifest sha256:3c87ccdc67513ff82d96394e208780d2b04502df3aee8a92026f30ff5bced
--> exporting config sha256:616ffdf049030b0a613662dbb65f47a1bc332ad6e04821c91683f6db7
--> exporting attestation manifest sha256:4d078708bcb35ee5790b4e7768338f7f4d4656ad0971aef462f42002ac72
--> exporting manifest list sha256:8d43d57fcf788938bf167427c5e39055d6f6ad15313648a95511b5bf7786c8c
--> pushing to docker.io/library/tfc_dam-backend:latest
--> unpacking to docker.io/library/tfc_dam-backend:latest
[backend] resolving provenance for metadata file
[frontend internal] load build definition from Dockerfile
[frontend internal] load
--> transferring dockerfile: 227B
[frontend internal] load metadata for docker.io/library/nginx:alpine
[frontend auth] library/nginx:pull token for registry-1.docker.io
[frontend auth] library/nginx:pull token for registry-1.docker.io
[frontend internal] load .dockerignore
[frontend internal] load build context
--> exporting layers
[frontend build 1/5] FROM docker.io/library/node:18@sha256:867be81f97d45cb7289a8ef0b128d23e8207412ebec4564441ed8cab8cc4ecd
--> resolve docker.io/library/node:18@sha256:867be81f97d45cb7289a8ef0b128d23e8207412ebec4564441ed8cab8cc4ecd
--> sha256:ef3c41c29e4d41d0e17287fc6b04324f2f4d065a09007f310735 447B / 447B
--> sha256:e1d0a5f55cde503d180849a2c641b5aff12cacfa7a3ec093986c82ea178c 1.25MB / 1.25MB
--> sha256:bfb3c08530ba4b1d7d7cef55cc32347e55d41408018cbf4a0bfc9a50b12fcd39 45.69MB / 45.69MB
--> sha256:68043c119c75902308a52f3a105c7f180a1c731f9a8a32f9d0808413 3.32kB / 3.32kB
--> sha256:c187b51b62e1d60ab36972b01f448ada945e97a5e137fc3180e0b7f90f 211.36MB / 211.36MB
--> sha256:6364a8518f54d31f8d8907f0071c7a793aa1aa8a4a3d77f4f43804ac8 24.01MB / 24.01MB
--> sha256:c031c000313ad37c509d5f9ee558a6d348bc3b6117f8c4a0732526 64.30MB / 64.30MB
--> sha256:cfe0542c035f802b6f9b0e549ac316cfb1dc8d8ac0d85ac728562ec9f2 48.49MB / 48.49MB
--> extracting sha256:cfe0542c035f802b6f9b0e549ac316cfb1dc8d8ac0d85ac728562ec9f2
--> extracting sha256:6364a8518f54d31f8d8907f0071c7a793aa1aa8a4a3d77f4f43804ac8
--> extracting sha256:ca13cad300b13ead3c745498459ee558a6d348bc3b6117f8c4a0732526
--> extracting sha256:c187b51b62e1d60ab36972b01f448ada945e97a5e137fc3180e0b7f90f
--> extracting sha256:68043c119c75902308a52f3a105c7f180a1c731f9a8a32f9d0808413
--> extracting sha256:bfb3c08530ba4b1d7d7cef55cc32347e55d41408018cbf4a0bfc9a50b12fcd39
--> extracting sha256:c1d0a5f55cde503d180849a2c641b5aff12cacfa7a3ec093986c82ea178c
--> extracting sha256:a03cf41c238c0ef42d0a1e71a875cf08d433472f2a6081a902077f3197a5
[frontend stage 1 1/2] FROM docker.io/library/nginx:alpine@sha256:65645c70b6a0601892a8b30808743208a18d02f3f17a54ef4b76fb02f2a10
--> resolve docker.io/library/nginx:alpine@sha256:65645c70b6a0601892a8b30808743208a18d02f3f17a54ef4b76fb02f2a10
--> sha256:30c203f0d010002a4a4067ca4095ac0b33eae0c0f179cc295400472a70 15.12MB / 15.12MB
--> sha256:1078075874efcc46724f17007240088a3080a4a0445feaf4f15093 1.21kB / 1.21kB
--> sha256:3a4a644b75651a26217f0000e1101a57208506c0d6c9a555a082a49a3102 1.40kB / 1.40kB
--> sha256:d9e902008397370b005445229963c346260ba9201900b2b3f25773 955B / 955B
--> sha256:81b0ed7ec678900cb7f1b47ee731c522f6d8a83201ec73c0bca1350f582948 402B / 402B
--> sha256:b046df22631975a0b73f0ec549790c45408214fc316ef915e438e5cd315 629B / 629B
--> sha256:61ca4773c802a9f08a312f0d0a31b6d713b3c3236c15700b1220f648074 1.40MB / 1.79MB
--> sha256:f182121f0c917a1f3d3a9d08011892101a032a93a388b78e990e0c2d5c870 3.64MB / 3.64MB
--> extracting sha256:f182121f0c917a1f3d3a9d08011892101a032a93a388b78e990e0c2d5c870
--> extracting sha256:61ca4773c802a9f08a312f0d0a31b6d713b3c3236c15700b1220f648074
--> extracting sha256:b046df22631975a0b73f0ec549790c45408214fc316ef915e438e5cd315
--> extracting sha256:d70b702400b3097a0b0a44545729963c346260ba9201900b2b3f25773
--> extracting sha256:81b0ed7ec678900cb7f1b47ee731c522f6d8a83201ec73c0bca1350f582948
--> extracting sha256:1078075874efcc46724f17007240088a3080a4a0445feaf4f15093
--> extracting sha256:3a4a644b75651a26217f0000e1101a57208506c0d6c9a555a082a49a3102
--> extracting sha256:39c2d5fd0810082aa0a0e7ca4e95ac90b33eae0c0f179cc295400472a70
[frontend internal] load build context
--> transferring context: 554.30MB
[frontend build 2/5] WORKDIR /app
[frontend build 3/5] COPY
[frontend build 4/5] RUN npm install
[frontend build 5/5] RUN npm run build
[frontend stage 1 2/2] COPY --from=build /app/build /usr/share/nginx/html
[frontend] exporting to image
...
```





6. Ejecutamos el script de la creación de la base de datos, en DBeaver.

ANEXO D.DOCUMENTACIÓN DE LA API

Usuarios:

- GET → (/usuario/getAll): Listar todos los usuarios.
- POST → (/usuario/add): Añadir un nuevo usuario.
- POST → (/usuario/addWithRol): Añade un usuario y le asigna un rol.
- POST → (/usuario/login): Autentica un usuario.
- POST → (/usuario/recuperar): Envía nueva contraseña por correo.
- PUT → (/usuario/update/{id}): Actualiza el usuario.
- DELETE → (/usuario/delete/{id}): Elimina un usuario.

Proyectos:

- GET → (/proyecto/dashboard): Listar todos los proyectos.
- POST → (/proyecto /add?usuariold=1): Crea un proyecto asociado a un usuario.
- POST → (/proyecto /asignarUsuario): Asigna un usuario a un proyecto.
- POST → (/proyecto /eliminarUsuario): Elimina un usuario de un proyecto.
- PUT → (/proyecto /update/{id}): Actualiza el proyecto.
- DELETE → (/proyecto /delete/{id}): Elimina un usuario.

Fases del Proyecto:

- GET → (/fases /nombres): Obtiene los nombres de las fases.
- POST → (/fases /add): Crea una nueva fase.
- PUT → (/fases /update/{id}): Actualiza una fase existente.
- DELETE → (/fases /delete/{id}): Elimina una fase.

Roles:

- GET → (/rol/getAll): Listar todos los roles.