Informe Taller de Proyecto

# TasksFlow – Gestión de Tareas Empresariales con Trazabilidad



Alumno: Diego Ignacio Alvial Arrepol

Carrera: Técnico en Programación y Análisis de Sistemas

Asignatura: Taller de Proyecto de la Especialidad

Docente: Víctor Valderrama Mora.

Sede: Concepción

Año: 2025

[TasksFlow – Gestión de Tareas Empresariales con Trazabilidad 1](#_Toc216631678)

[Resumen Ejecutivo 3](#_Toc216631679)

[Problema y Propuesta de Solución 4](#_Toc216631680)

[Objetivos SMART 4](#_Toc216631681)

[Elementos de Innovación 6](#_Toc216631682)

[Reflexión - La Catedral y el Bazar 6](#_Toc216631683)

[Licencia del Proyecto GPL 6](#_Toc216631684)

[Arquitectura Técnica del Sistema 7](#_Toc216631685)

[Diagrama de Procesos / DFD 8](#_Toc216631686)

[Historias de Usuario 9](#_Toc216631687)

[Requerimientos del Sistema 15](#_Toc216631688)

[Metodologías de Desarrollo Aplicadas 28](#_Toc216631689)

[Casos de Uso del Sistema TasksFlow 32](#_Toc216631690)

[Diagramas generales del sistema 46](#_Toc216631691)

[MER Modelo Entidad Relación 54](#_Toc216631692)

[Diseño físico de la base de datos 55](#_Toc216631693)

[Wireframes del Sistema 61](#_Toc216631694)

[Wireflows del Sistema 65](#_Toc216631695)

[Plan de Pruebas 69](#_Toc216631696)

[Manual de Usuario 70](#_Toc216631697)

[Conclusiones 72](#_Toc216631698)

## Resumen Ejecutivo

TasksFlow es una aplicación web orientada a la gestión de tareas empresariales con trazabilidad, diseñada para organizaciones que requieren controlar, coordinar y auditar el trabajo diario de sus equipos de manera estructurada. El sistema aborda el problema común de la fragmentación en la gestión de tareas —realizada habitualmente mediante correos, planillas o mensajería informal— que dificulta el seguimiento del avance, la coordinación entre áreas y la rendición de cuentas.

La solución propuesta se materializa en una plataforma tipo **Kanban**, organizada por **empresa, áreas, equipos y roles jerárquicos** (Root, Admin, Supervisor y User), que permite crear proyectos, asignar tareas, moverlas entre estados y registrar cada avance mediante un **historial de acciones inspirado en Git**, denominado “commits de tareas”. Esta funcionalidad aporta un nivel adicional de trazabilidad, permitiendo conocer qué se hizo, quién lo hizo y cuándo se realizó cada acción.

Desde el punto de vista técnico, TasksFlow fue desarrollado utilizando **Node.js y Express** para el backend, **MySQL** como sistema gestor de base de datos relacional, y una interfaz web basada en **EJS, JavaScript y Tailwind CSS**. La arquitectura del sistema sigue un modelo MVC simple, con control de acceso por roles, segmentación por empresa (company\_id) y prácticas de desarrollo ágil mediante Kanban personal y principios de Extreme Programming (XP).

El proyecto cumple con los objetivos académicos del Taller de Proyecto de la Especialidad, integrando análisis, diseño, implementación funcional, pruebas y documentación, y constituye un prototipo sólido, escalable y alineado con necesidades reales del entorno empresarial.

## Problema y Propuesta de Solución

En muchas empresas y organizaciones, la gestión de tareas diarias se realiza de forma descentralizada, mediante hojas de cálculo, correos electrónicos o canales de mensajería dispersos. Esta fragmentación dificulta el control del avance, la trazabilidad de las acciones y la coordinación entre las diferentes áreas de trabajo. Supervisores y trabajadores suelen carecer de una herramienta centralizada que les permita planificar, asignar y dar seguimiento a las actividades de manera visual y estructurada.

Propuesta:  
Desarrollar una aplicación web tipo Kanban para la gestión de tareas empresariales, organizada por áreas, supervisores y trabajadores. El sistema permitirá crear tableros de tareas por área, asignar responsables, registrar avances y mantener un historial de cambios y acciones inspirado en las buenas prácticas de Git (como versiones o estados de tareas). De este modo, se busca mejorar la organización interna, la comunicación y la transparencia en el seguimiento de las actividades cotidianas dentro de la empresa.

## Objetivos SMART

Objetivo General

Gestionar mediante una aplicación web tipo Kanban la administración centralizada de tareas en empresas, organizada por áreas, supervisores y trabajadores, que permita asignar, registrar y dar seguimiento a actividades cotidianas con trazabilidad de acciones inspirada en Git.

Objetivos Específicos

|  |  |
| --- | --- |
| S (Específico) | Crear una app web Kanban empresarial con jerarquía de usuarios. |
| M (Medible) | Entregar del sistema funcional y documentado. |
| A (Alcanzable) | Programar con Node.js, Express y MySQL2. |
| R (Relevante) | Mejorar la organización, comunicación y control interno. |
| T (Tiempo) | Entregar un plazo de 12 semanas. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | S | M | A | R | T |
| Específico 1 | Levantar y documentar los requerimientos funcionales y no funcionales del sistema, definiendo roles y flujos de trabajo. | | | | |
| Definir requerimientos y estructura de roles. | Documento de requerimientos validado. | Factible mediante entrevistas y análisis interno. | Es la base para el diseño del sistema. | Durante las 2 primeras semanas. |
| Específico 2 | Diseñar la arquitectura del sistema y el modelo de base de datos relacional. | | | | |
| Elaborar diagramas UML/DFD y modelo ER. | Entregables técnicos revisados. | Realizable con herramientas de modelado. | Permite una implementación estructurada. | En la semana 3. |
| Específico 3 | Implementar una API REST en Node.js con Express y MySQL2 para la gestión de usuarios, áreas y tareas. | | | | |
| Desarrollar endpoints funcionales. | Verificación mediante pruebas de API. | Totalmente alcanzable con tu stack actual. | Proporciona la lógica principal del sistema. | En un periodo de 4 semanas. |
| Específico 4 | Desarrollar la interfaz web responsiva con tablero Kanban integrado al backend. | | | | |
| Construir la UI de interacción con tareas. | Funcionalidad validada mediante pruebas de usuario. | Realizable con frameworks front-end o HTML/CSS. | Es la interfaz visual principal del sistema. | En 3 semanas. |
| Específico 5 | Integrar el registro de acciones e historial de cambios inspirado en Git. | | | | |
| Implementar módulo de trazabilidad. | Validación mediante logs y visualización de historial. | Viable mediante registro de acciones en BD. | Aporta innovación y valor diferencial. | En 2 semanas. |
| Específico 6 | Realizar pruebas funcionales, documentación técnica y despliegue básico del sistema. | | | | |
| Probar y documentar todo el proyecto. | Entrega de informe y sistema operativo. | Alcanzable con herramientas de testing y documentación. | Asegura la calidad y cierre del proyecto. | En la última semana (semana 12). |

## Elementos de Innovación

El proyecto propone una innovación funcional al trasladar principios de control de versiones y trazabilidad, propios de entornos de desarrollo de software (como Git), al ámbito de la gestión de tareas empresariales. En lugar de centrarse en el control de código fuente, la aplicación aplica el concepto de historial de cambios y registro de acciones para las tareas cotidianas de los trabajadores. Esto permite que cada tarea cuente con un seguimiento estructurado y transparente, facilitando la comunicación entre supervisores y empleados.  
  
Además, el sistema se distingue por su estructura jerárquica basada en áreas, supervisores y trabajadores, lo que permite adaptar la plataforma a diferentes tipos de organizaciones. Esta combinación de un tablero Kanban con trazabilidad inspirada en Git representa una propuesta innovadora dentro del ámbito de la gestión de procesos internos.

## Reflexión - La Catedral y el Bazar

La reflexión 'La Catedral y el Bazar' plantea dos modelos de desarrollo de software: la 'Catedral', caracterizada por la planificación centralizada y el control jerárquico, y el 'Bazar', basado en la colaboración abierta y la mejora continua por parte de múltiples participantes.  
  
El enfoque de este proyecto combina elementos de ambos modelos. Desde la perspectiva de la 'Catedral', se sigue una planificación estructurada y una jerarquía de roles (áreas, supervisores, trabajadores), que asegura control y claridad en las responsabilidades. Sin embargo, también incorpora la filosofía del 'Bazar', ya que fomenta la colaboración, la comunicación abierta y la mejora continua mediante el uso de tableros Kanban y el registro transparente de acciones.  
  
En síntesis, el proyecto busca equilibrar la estructura organizacional (Catedral) con la flexibilidad y participación (Bazar), logrando así un entorno digital más colaborativo dentro de la empresa.

## Licencia del Proyecto GPL

El proyecto se distribuirá bajo la Licencia Pública General de GNU (GPL v3), una de las licencias más reconocidas del software libre. Esta licencia garantiza que el código fuente del programa permanezca abierto y disponible para todos los usuarios, quienes podrán usar, estudiar, modificar y redistribuir el software, siempre y cuando mantengan los mismos términos de libertad en las versiones derivadas.  
  
La elección de la licencia GPL se justifica porque promueve la colaboración, la transparencia y el desarrollo comunitario, asegurando que cualquier mejora o adaptación del sistema también se comparta bajo la misma licencia.

De esta manera, el proyecto se alinea con los valores del software libre y con la filosofía de apertura y mejora continua que inspiran su desarrollo.

## Arquitectura Técnica del Sistema

TasksFlow se implementa bajo una arquitectura **cliente–servidor**, separando claramente la capa de presentación, la lógica de negocio y el acceso a datos. Esta separación mejora la mantenibilidad, la escalabilidad y el control de seguridad del sistema.

Arquitectura lógica

**Capa Cliente (Frontend):**

* Interfaz web desarrollada con **EJS**, **HTML**, **JavaScript** y **Tailwind CSS**.
* Vistas diferenciadas según rol: Root, Admin, Supervisor y User.
* Interacción dinámica mediante fetch y eventos JavaScript para operaciones como:
  + Cambio de estado en Kanban
  + Creación y edición de tareas y proyectos
  + Registro de commits

**Capa Servidor (Backend):**

* Servidor **Node.js con Express**.
* Organización del código en patrón **MVC**:
  + **Routes:** definición de endpoints HTTP y control de acceso.
  + **Controllers:** lógica de negocio y validaciones.
  + **Models:** acceso a datos mediante consultas SQL estructuradas.
* Middlewares de:
  + Autenticación (requireAuth)
  + Autorización por rol
  + Validación de datos

**Capa de Datos:**

* **MySQL 8+** como SGBD.
* Base de datos normalizada con claves foráneas y control de integridad.
* Aislamiento lógico por empresa mediante company\_id.

**Seguridad y control de acceso**

* Contraseñas almacenadas mediante **hash seguro (Argon2)**.
* Sesiones controladas por backend.
* Restricción de rutas y vistas según rol.
* Separación estricta de datos entre empresas.

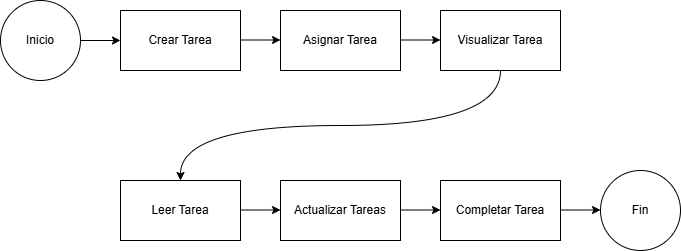
**Despliegue**

El sistema está preparado para ejecutarse en:

* Entorno local (Node.js + MySQL).
* Servicios en la nube como **Azure App Service**, utilizando variables de entorno para la configuración sensible.

## Diagrama de Procesos / DFD

El siguiente diagrama representa de manera general el flujo de procesos del sistema Kanban empresarial. En esta primera versión, el modelo describe cómo las tareas son creadas, asignadas y actualizadas dentro del flujo de trabajo jerárquico:  
  
1. El \*\*Supervisor de Área\*\* crea y organiza tableros de tareas para su equipo.  
2. Cada \*\*Trabajador\*\* visualiza las tareas asignadas y actualiza su estado (To Do, In Progress, Done).  
3. El sistema registra cada cambio en una base de datos y actualiza el historial de acciones.  
4. Los \*\*Administradores o Supervisores\*\* pueden visualizar reportes de avance y trazabilidad.



## Historias de Usuario

HU-01 – Registro y gestión de empresas (Root)

**ID:** HU-01  
**Nombre:** Administración de empresas  
**Actor principal:** Root  
**Prioridad:** Alta

**Descripción**

**Como** administrador global del sistema (Root),  
**quiero** registrar, editar y desactivar empresas dentro de la plataforma,  
**para** permitir que diferentes organizaciones utilicen TasksFlow con espacios de datos independientes por company\_id.

**Criterios de aceptación**

* **CA-01:** El Root puede crear una empresa ingresando nombre, descripción y estado.
* **CA-02:** Cada empresa queda asociada a un company\_id único.
* **CA-03:** El Root puede asignar un usuario administrador inicial a la empresa.
* **CA-04:** El Root puede activar o desactivar empresas existentes.
* **CA-05:** Los datos de una empresa sólo pueden ser vistos por el Root, no por otros roles.

HU-02 – Gestión de usuarios por empresa (Root/Admin)

**ID:** HU-02  
**Nombre:** Administración de usuarios  
**Actor principal:** Root / Admin  
**Prioridad:** Alta

**Descripción**

**Como** administrador (global o de empresa),  
**quiero** crear, editar y desactivar usuarios dentro de una empresa,  
**para** mantener el control de acceso por rol, área y compañía.

**Criterios de aceptación**

* **CA-01:** El Root puede administrar usuarios de todas las empresas.
* **CA-02:** El Admin solo puede administrar usuarios de su empresa (company\_id).
* **CA-03:** El formulario permite asignar rol: admin, supervisor o user.
* **CA-04:** El sistema valida que el correo no esté repetido.
* **CA-05:** Los usuarios desactivados no pueden iniciar sesión.

HU-03 – Gestión de áreas (Admin)

**ID:** HU-03  
**Nombre:** Configuración de áreas  
**Actor principal:** Admin  
**Prioridad:** Media

**Descripción**

**Como** administrador de empresa,  
**quiero** crear y gestionar las áreas internas (Operaciones, Logística, etc.),  
**para** organizar los equipos y supervisores según la estructura de la empresa.

**Criterios de aceptación**

* **CA-01:** El Admin puede registrar áreas asociadas a su company\_id.
* **CA-02:** Cada área puede tener un estado (activa/inactiva).
* **CA-03:** Las áreas pueden verse en selectores de creación de usuarios, equipos y proyectos.

HU-04 – Gestión de equipos de trabajo (Admin)

**ID:** HU-04  
**Nombre:** Crear y gestionar equipos  
**Actor principal:** Admin  
**Prioridad:** Media

**Descripción**

**Como** administrador de empresa,  
**quiero** crear equipos de trabajo y asignar miembros,  
**para** organizar mejor los proyectos y distribuir la carga laboral.

**Criterios de aceptación**

* **CA-01:** El Admin puede crear equipos con nombre, tipo y estado.
* **CA-02:** El Admin puede agregar o remover miembros del equipo.
* **CA-03:** Los miembros deben pertenecer a la misma empresa.
* **CA-04:** Los equipos aparecen en listados y selectores para asignación de tareas.

HU-05 – Creación y gestión de proyectos (Admin/Supervisor/User)

**ID:** HU-05  
**Nombre:** Registrar proyectos  
**Actor principal:** Admin / Supervisor / User  
**Prioridad:** Alta

**Descripción**

**Como** usuario con rol permitido,  
**quiero** crear proyectos dentro de mi empresa,  
**para** organizar las tareas y actividades según área o iniciativa personal.

**Criterios de aceptación**

* **CA-01:** Admin y Supervisor pueden crear proyectos para su empresa y área.
* **CA-02:** Los usuarios pueden crear proyectos personales asociados solo a ellos.
* **CA-03:** El sistema almacena creator\_id, company\_id, area\_id y estado.
* **CA-04:** Los proyectos pueden ser editados (nombre, fechas, estado).
* **CA-05:** Los proyectos se muestran en los selectores de tareas y en los dashboards.

HU-06 – Crear tareas para equipos y áreas (Supervisor/Admin)

**ID:** HU-06  
**Nombre:** Registrar tareas de equipo  
**Actor principal:** Supervisor / Admin  
**Prioridad:** Alta

**Descripción**

**Como** supervisor o administrador,  
**quiero** crear tareas asignadas a usuarios de mi área,  
**para** planificar, organizar y controlar el avance del trabajo del equipo.

**Criterios de aceptación**

* **CA-01:** El creador puede asignar uno o más usuarios responsables.
* **CA-02:** Cada tarea tiene estado inicial pending.
* **CA-03:** La tarea se asocia a un proyecto (opcional).
* **CA-04:** La tarea se registra en la tabla tasks respetando el company\_id.
* **CA-05:** Las tareas se muestran en el dashboard del supervisor y en el Kanban.

HU-07 – Tareas personales (User)

**ID:** HU-07  
**Nombre:** Crear tareas personales  
**Actor principal:** User  
**Prioridad:** Media

**Descripción**

**Como** usuario de la plataforma,  
**quiero** crear tareas personales con o sin proyecto,  
**para** organizar mis actividades individuales dentro de la empresa.

**Criterios de aceptación**

* **CA-01:** El usuario puede marcar la tarea como personal (is\_personal = 1).
* **CA-02:** La tarea personal solo es visible para su creador.
* **CA-03:** La tarea sigue el flujo Kanban del usuario.
* **CA-04:** Puede asociarse opcionalmente a un proyecto personal.

HU-08 – Tablero Kanban por rol

**ID:** HU-08  
**Nombre:** Visualizar tablero Kanban  
**Actor principal:** User / Supervisor / Admin  
**Prioridad:** Alta

**Descripción**

**Como** usuario según mi rol,  
**quiero** visualizar las tareas en un tablero Kanban,  
**para** comprender el estado del trabajo y avanzar las tareas entre columnas.

**Criterios de aceptación**

* **CA-01:** El Kanban muestra las columnas: pending, in\_progress, review, done.
* **CA-02:** Cada rol ve solo las tareas correspondientes a su alcance:
  + Admin: tareas de la empresa.
  + Supervisor: tareas de su área/equipo.
  + User: tareas asignadas + personales.
* **CA-03:** El cambio de columna actualiza el estado de la tarea.

HU-09 – Historial tipo Git (commits)

**ID:** HU-09  
**Nombre:** Registrar commits de tarea  
**Actor principal:** User / Supervisor / Admin  
**Prioridad:** Media

**Descripción**

**Como** usuario asignado a una tarea,  
**quiero** registrar un commit con mensaje y actualización de estado,  
**para** dejar trazabilidad del avance y decisiones sobre la tarea.

**Criterios de aceptación**

* **CA-01:** Cada commit guarda: autor, mensaje, estado previo y estado nuevo.
* **CA-02:** Los commits se listan en un historial ordenado.
* **CA-03:** Un commit puede opcionalmente cambiar el estado de la tarea.
* **CA-04:** El historial es visible para el supervisor y el usuario asignado.

HU-10 – Autenticación y control de acceso

**ID:** HU-10  
**Nombre:** Login por rol  
**Actor principal:** Todos los roles  
**Prioridad:** Alta

**Descripción**

**Como** usuario del sistema,  
**quiero** iniciar sesión con mi correo y contraseña,  
**para** acceder solo a las funciones habilitadas por mi rol y empresa.

**Criterios de aceptación**

* **CA-01:** El sistema valida credenciales con contraseñas hasheadas.
* **CA-02:** Se cargan rol, área y empresa del usuario.
* **CA-03:** Cada vista (root, admin, supervisor, user) se habilita o restringe según rol.
* **CA-04:** Un usuario no puede ver información de otra empresa.

## Requerimientos del Sistema

Este capítulo presenta los requerimientos funcionales (RF) y no funcionales (RNF) del sistema TasksFlow, definidos a partir del análisis de objetivos específicos, arquitectura preliminar y flujos de trabajo establecidos.

Requerimientos funcionales del sistema (entrada, proceso y salida)

A continuación, se describen los principales requerimientos funcionales de *TasksFlow* organizados como procesos con sus **entradas**, **transformaciones internas** y **salidas**, considerando los cuatro roles del sistema (Root, Admin, Supervisor y Usuario) y el contexto multiempresa, donde cada operación se encuentra asociada a un company\_id independiente.

RF-01 – Gestión de empresas y administradores (Root)

**Actor principal:** Root (administrador del sitio).  
**Actores secundarios:** Admin (de cada empresa).

* **Entradas**
  + Datos de la empresa: nombre, descripción, estado (activa/inactiva).
  + Datos del administrador principal de la empresa: nombre, correo, rol admin, credenciales de acceso.
  + Parámetros de configuración inicial (por ejemplo, si la empresa está habilitada para operar en el sistema).
* **Proceso**
  + El usuario Root accede al panel de administración global.
  + Registra una nueva empresa, asociándola a un company\_id único.
  + Crea o asigna un usuario con rol admin para esa empresa.
  + Actualiza o desactiva empresas cuando sea necesario (por ejemplo, baja del servicio).
* **Salidas**
  + Empresa registrada en la base de datos, visible sólo para los usuarios pertenecientes a ese company\_id.
  + Usuario administrador asociado a la empresa, habilitado para gestionar áreas, equipos, proyectos y tareas de su organización.
  + Listado actualizado de empresas disponibles para el Root, con sus estados.

RF-02 – Gestión de usuarios y roles por empresa

**Actores principales:** Root, Admin.  
**Actores secundarios:** Supervisor (consulta parcial).

* **Entradas**
  + Datos personales del usuario (nombre, correo, teléfono, etc.).
  + Rol asignado (root, admin, supervisor, user).
  + Empresa (company\_id) y, cuando aplique, área (area\_id) y supervisor asociado (manager\_id).
  + Credenciales de acceso (contraseña encriptada con hash Argon2 en el backend).
* **Proceso**
  + Root o Admin accede al módulo de gestión de usuarios.
  + Crea, edita o desactiva usuarios, definiendo el rol y la vinculación con la empresa y el área correspondiente.
  + El sistema valida la unicidad del correo y aplica políticas de seguridad (hash de contraseña).
  + Se restringe el acceso a la gestión de usuarios según el rol:
    - Root: puede ver/administrar usuarios de todas las empresas.
    - Admin: puede ver/administrar sólo usuarios de su company\_id.
* **Salidas**
  + Usuario creado o actualizado en la tabla users, con rol y empresa asociados.
  + Listados filtrados de usuarios por empresa y/o área, usados luego en los formularios de asignación de tareas y equipos.
  + Control de acceso coherente con el rol al momento de iniciar sesión y navegar por el sistema (seguridad y segmentación por empresa).

RF-03 – Gestión de áreas por empresa

**Actores principales:** Admin.  
**Actores secundarios:** Supervisor.

* **Entradas**
  + Datos del área: nombre, descripción, estado (activa/inactiva).
  + Empresa a la que pertenece el área (company\_id).
* **Proceso**
  + El Admin abre el panel de administración de su empresa.
  + Registra nuevas áreas o actualiza áreas existentes (por ejemplo, Operaciones, Mantenimiento, Ventas).
  + Opcionalmente, asigna supervisores responsables de cada área.
* **Salidas**
  + Áreas registradas y visibles sólo dentro de la empresa correspondiente.
  + Listado de áreas activo que se utiliza en:
    - Asignación de usuarios a un área.
    - Filtrado de proyectos y tareas por area\_id.
    - Dashboard del supervisor asociado a esa área.

RF-04 – Gestión de equipos (teams) dentro de la empresa

**Actores principales:** Admin.  
**Actores secundarios:** Supervisor.

* **Entradas**
  + Datos del equipo: nombre, descripción, tipo, estado.
  + Empresa (company\_id) y área a la que pertenece el equipo.
  + Usuarios que serán miembros del equipo (lista de user\_id).
* **Proceso**
  + El Admin crea equipos de trabajo asociados a una empresa y, opcionalmente, a un área.
  + Agrega o quita miembros del equipo mediante las rutas /api/teams/:id/members/add y /remove (sólo admin/root).
  + El sistema actualiza la tabla intermedia de miembros del equipo, respetando el company\_id.
* **Salidas**
  + Equipos registrados con sus miembros asociados.
  + Equipos disponibles en selectores y filtros para:
    - Asignar tareas a miembros específicos.
    - Analizar carga de trabajo por equipo en los dashboards tácticos.

RF-05 – Gestión de proyectos por empresa, área y creador

**Actores principales:** Admin, Supervisor, Usuario (proyectos personales).  
**Actores secundarios:** Root (consulta global).

* **Entradas**
  + Datos del proyecto: nombre, descripción, estado (active, inactive), fechas de inicio/fin.
  + Empresa (company\_id) y, cuando aplica, área (area\_id) asociada.
  + Usuario que crea el proyecto (creator\_id), que puede ser Admin, Supervisor o User.
* **Proceso**
  + El sistema expone una API de proyectos accesible para roles admin, supervisor, root y user (según permisos).
  + Se permite crear proyectos de área (por Admin/Supervisor) y proyectos personales (por User), siempre dentro de una empresa.
  + Se listan proyectos por:
    - Empresa y área (para supervisores y admin).
    - Proyectos donde el usuario actual es creador o participante (para Users y Supervisores).
  + Se actualizan proyectos (nombre, estado, fechas) usando la ruta /api/projects/update.
* **Salidas**
  + Proyectos registrados y visibles según el rol:
    - Admin: todos los proyectos de la empresa.
    - Supervisor: proyectos del área y proyectos creados por él.
    - User: proyectos personales y donde participa.
  + Listas de proyectos utilizadas en:
    - Selectores de creación/edición de tareas.
    - Filtros del tablero Kanban (por proyecto).
    - Reportes de carga y avance por proyecto.

RF-06 – Gestión de tareas de equipo / área / compañía

**Actores principales:** Supervisor, Admin.  
**Actores secundarios:** User (como asignado).

* **Entradas**
  + Datos de la tarea: título, descripción, prioridad, fecha límite (deadline), estado inicial (pending), visibilidad (team, area, company), indicador de si es personal o no.
  + Proyecto asociado (opcional).
  + Usuarios asignados a la tarea (relación N:M mediante task\_assignments).
  + Empresa (company\_id) y área derivada del supervisor/creador.
* **Proceso**
  + El Supervisor o Admin crea tareas para su área/equipo a través de formularios específicos.
  + El backend registra la tarea en tasks y sus asignaciones en task\_assignments.
  + Se actualiza el estado de la tarea (pending, in\_progress, review, done) mediante acciones desde el tablero Kanban o desde los commits.
  + El sistema actualiza indicadores de:
    - Tareas abiertas, en progreso y vencidas.
    - Carga por usuario y por equipo.
* **Salidas**
  + Tareas registradas y visibles en:
    - Dashboard del Supervisor (vista táctica con KPIs de equipo y alertas).
    - Tablero Kanban del Supervisor (tareas por estado).
  + Notificación visual al trabajador asignado (la tarea aparece en su tablero de usuario).
  + Estadísticas agregadas por estado, por usuario y por fecha de vencimiento, utilizadas en alertas tácticas y planificación.

RF-07 – Gestión de tareas personales del usuario

**Actor principal:** User.  
**Actores secundarios:** Supervisor (consulta parcial).

* **Entradas**
  + Datos de la tarea personal: título, descripción, prioridad, fecha límite.
  + Proyecto asociado (opcional) o “sin proyecto” (tarea suelta).
  + Indicador de tarea personal (is\_personal) y visibilidad privada.
* **Proceso**
  + El usuario abre el modal de “Nueva tarea personal” y completa el formulario.
  + El sistema crea la tarea vinculada al user.id como creador y asignado.
  + El usuario actualiza el estado de sus tareas directamente desde el tablero Kanban personal (pending → in\_progress → review → done).
* **Salidas**
  + Lista de tareas personales visible en:
    - Dashboard de usuario (resumen por estado: pendientes, en progreso, en revisión y completadas).
    - Tablero Kanban de usuario, con filtro por proyecto / sin proyecto.
  + Indicadores individuales de avance que ayudan al usuario a organizar su trabajo diario.

RF-08 – Tablero Kanban por rol (Admin, Supervisor, User)

**Actores principales:** Admin, Supervisor, User.  
**Actor secundario:** Root (consulta).

* **Entradas**
  + Tareas registradas y filtradas según rol:
    - Admin: tareas a nivel empresa.
    - Supervisor: tareas de su área/equipo.
    - User: tareas donde participa o tareas personales.
  + Estado de cada tarea y proyecto asociado, cuando aplica.
* **Proceso**
  + El usuario activa el toggle de vista Kanban en la interfaz, que alterna entre Dashboard y Kanban según su rol.
  + El sistema muestra columnas de estado (pending, in\_progress, review, done), con tarjetas arrastrables que representan tareas.
  + Se aplican filtros por proyecto o por “tareas sin proyecto”.
  + Cambios de estado pueden disparar registros de commits y actualizaciones de métricas.
* **Salidas**
  + Visualización clara del flujo de trabajo por columna de estado.
  + Distribución de tareas por usuario, por proyecto y por estado.
  + Base visual para la toma de decisiones tácticas (supervisor) y operativas (usuario).

RF-09 – Historial de acciones tipo Git (commits de tareas)

**Actores principales:** User, Supervisor.  
**Actores secundarios:** Admin (revisión), Root (auditoría).

* **Entradas**
  + Selección de una tarea específica.
  + Mensaje de commit que describe el avance, bloqueo o instrucción.
  + Estado actual de la tarea y estado destino (opcional) si se desea cambiar (to\_status).
  + Usuario que realiza el commit (author\_id).
* **Proceso**
  + El usuario abre el modal de commits de la tarea.
  + Registra un nuevo commit, que se guarda en task\_commits con información de mensaje, estado previo, estado posterior, fecha y autor.
  + El sistema actualiza, si corresponde, el estado de la tarea.
  + Se recupera el historial completo de commits para esa tarea (ordenado por fecha) cuando el modal es abierto.
* **Salidas**
  + Historial detallado de cambios y decisiones para cada tarea, visible tanto para el usuario como para el supervisor.
  + Línea de tiempo que permite revisar:
    - Qué se hizo.
    - Cuándo se hizo.
    - Quién lo hizo.
  + Evidencia de trazabilidad para informes y revisiones posteriores, alineada con el objetivo de “historial de acciones tipo Git”.

RF-10 – Autenticación y control de acceso por roles

**Actores principales:** Todos los roles (Root, Admin, Supervisor, User).

* **Entradas**
  + Credenciales de acceso: correo y contraseña.
  + Rol asociado al usuario.
  + company\_id y area\_id cuando correspondan.
* **Proceso**
  + El usuario ingresa sus credenciales en el formulario de login.
  + El backend valida la contraseña encriptada y recupera información de rol, empresa, área y permisos.
  + El sistema genera una sesión de usuario y restringe el acceso a rutas según rol:
    - Ejemplos:
      * /admin: sólo admin y root.
      * /supervisor: supervisor y root.
      * /user: user y root.
      * APIs de proyectos/tareas limitadas por rol y company\_id.
* **Salidas**
  + Acceso autorizado a la vista correspondiente (Root, Admin, Supervisor o User).
  + Menús, dashboards, modales y funciones habilitadas o restringidas según el rol.
  + Protección de la información multiempresa: un usuario nunca ve datos de otra empresa

Requerimientos No Funcionales (RNF)

A continuación, se presentan los requerimientos no funcionales de TasksFlow, clasificados según las áreas que exige el instructivo: **usabilidad, seguridad, rendimiento, disponibilidad, interoperabilidad, mantenibilidad y portabilidad**.

RNF-01 – Usabilidad

**Descripción:**  
El sistema debe ser intuitivo, coherente y fácil de usar para usuarios con distintos niveles de conocimiento tecnológico.

**Criterios:**

* RNF-01.1: La interfaz debe ser responsive y adaptarse a pantallas desktop, tablet y móvil.
* RNF-01.2: Los cuatro roles (Root, Admin, Supervisor, User) deben ver únicamente las funciones relevantes para ellos (mínima complejidad visual).
* RNF-01.3: Los formularios deben incluir placeholders, validaciones y mensajes de error claros.
* RNF-01.4: Cambiar entre **Dashboard/Kanban** debe ser inmediato y sin recargar la página.
* RNF-01.5: Para tareas y proyectos, los modales deben abrirse rápido, con inputs prellenados cuando corresponda.

RNF-02 – Seguridad

**Descripción:**  
El sistema debe proteger la información de cada empresa y controlar el acceso según roles.

**Criterios:**

* RNF-02.1: Las contraseñas deben almacenarse usando hashing seguro (Argon2 o).
* RNF-02.2: Se debe implementar middleware de autenticación y autorización por rol.
* RNF-02.3: Cada usuario solo debe acceder a información de su company\_id.
* RNF-02.4: Las rutas sensibles deben requerir sesión activa (requireAuth).
* RNF-02.5: Los formularios deben validar datos en frontend y backend para evitar inyección SQL o datos maliciosos.
* RNF-02.6: No almacenar datos sensibles del usuario en el cliente (solo la sesión).

RNF-03 – Rendimiento

**Descripción:**  
El sistema debe ofrecer tiempos de respuesta adecuados incluso con múltiples usuarios dentro de una misma empresa.

**Criterios:**

* RNF-03.1: Las consultas SQL deben usar índices en columnas críticas (company\_id, area\_id, project\_id).
* RNF-03.2: El sistema debe cargar dashboards en menos de 2 segundos en condiciones normales.
* RNF-03.3: Los cambios de estado en el Kanban deben impactar solo la tarea modificada, sin recargar toda la vista.
* RNF-03.4: Las tareas y proyectos deben cargarse paginadas o filtradas cuando superan cierto volumen (opcional para futuras versiones).

RNF-04 – Disponibilidad

**Descripción:**  
El sistema debe estar disponible y operativo para los usuarios durante la jornada laboral de la empresa.

**Criterios:**

* RNF-04.1: La aplicación debe poder desplegarse en servicios de nube como Azure App Service (según tu implementación).
* RNF-04.2: El sistema debe manejar errores comunes (404, 500) mostrando mensajes amigables.
* RNF-04.3: Fallas en una empresa no deben afectar a las demás (aislamiento por company\_id).
* RNF-04.4: Los datos deben mantenerse accesibles incluso tras reinicios del servidor (persistencia en MySQL).

RNF-05 – Mantenibilidad

**Descripción:**  
El sistema debe mantener una estructura clara que facilite corregir errores y agregar nuevas funcionalidades.

**Criterios:**

* RNF-05.1: El código debe estar organizado en capas (rutas, controladores, modelos, vistas).
* RNF-05.2: El sistema debe usar control de versiones mediante GitHub y estructura de commits clara.
* RNF-05.3: Se deben documentar los modelos y rutas principales.
* RNF-05.4: La base de datos debe ser escalable para agregar nuevas entidades sin romper relaciones ya existentes.
* RNF-05.5: Los estilos y scripts deben estar modularizados (por ejemplo, funciones JS separadas para cada vista).

RNF-06 – Interoperabilidad

**Descripción:**  
El sistema debe integrarse correctamente con otros servicios cuando sea necesario.

**Criterios:**

* RNF-06.1: La API debe entregar y recibir datos en formato JSON.
* RNF-06.2: El sistema debe permitir integración futura con servicios de notificaciones, correo u otros microservicios.
* RNF-06.3: Debe ser posible conectar el front-end con la API por medio de fetch, AJAX o frameworks futuros.

RNF-07 – Portabilidad

**Descripción:**  
El sistema debe poder ejecutarse en entornos de desarrollo o producción con cambios mínimos.

**Criterios:**

* RNF-07.1: El proyecto debe funcionar en cualquier entorno compatible con Node.js 18+ y MySQL 8+.
* RNF-07.2: Las variables sensibles deben estar en .env.
* RNF-07.3: El sistema debe ser desplegable tanto en Windows como en Linux (Azure App Service o VPS).
* RNF-07.4: El front-end basado en EJS/Tailwind debe funcionar en navegadores modernos (Chrome, Edge, Firefox).

## Metodologías de Desarrollo Aplicadas

El desarrollo del sistema *TasksFlow* se realizó mediante una **metodología ágil híbrida**, combinando elementos de **Lean Personal Kanban** y prácticas seleccionadas de **Extreme Programming (XP)**. Esta metodología fue elegida considerando el contexto académico, el alcance del proyecto y el hecho de que se trata de un desarrollo individual con entregas iterativas.

A continuación, se presenta cada metodología en formato estructurado.

MD-01 – Lean Personal Kanban

**Nombre:** Lean Personal Kanban  
**Tipo:** Metodología ágil visual  
**Aplicación:** Gestión del flujo de trabajo durante todo el proyecto  
**Propósito:** Mantener visibilidad, ritmo constante y control de carga de trabajo

**Descripción**

Lean Personal Kanban se utilizó para **organizar, visualizar y priorizar** las tareas del proyecto mediante un tablero Kanban personal. El tablero contaba con las columnas:

* **To Do**
* **In Progress**
* **Done**

Esta visualización permitió gestionar el progreso de manera clara, reduciendo sobrecarga y mejorando el enfoque.

**Principios aplicados**

* **MD-01.1 – Visualización del flujo:**  
  Todas las tareas del proyecto fueron registradas y movidas entre columnas, entregando un estado real del avance.
* **MD-01.2 – Límite de trabajo en progreso (WIP):**  
  Se evitó trabajar en demasiadas tareas al mismo tiempo, disminuyendo errores y mejorando la concentración.
* **MD-01.3 – Priorización continua:**  
  Las tareas fueron reorganizadas frecuentemente según entregas, dependencias técnicas y exigencias académicas.

**Beneficios obtenidos**

* Mayor claridad en hitos, entregables y pendientes.
* Detección temprana de bloqueos o problemas técnicos.
* Mejor gestión del tiempo y del ritmo de avance.
* Flujo de trabajo estable y visual durante todo el proyecto.

MD-02 – Extreme Programming (XP)

**Nombre:** Extreme Programming (XP)  
**Tipo:** Metodología ágil orientada a calidad técnica  
**Aplicación:** Desarrollo backend y procesos iterativos  
**Propósito:** Mantener un código simple, estable, escalable y fácil de mejorar

**Descripción**

Se adoptaron prácticas de XP adaptadas a un proyecto individual en Node.js/Express. Estas prácticas permitieron mantener calidad técnica constante mientras se avanzaba en funcionalidades.

**Prácticas XP aplicadas**

* **MD-02.1 – Simplicidad técnica:**  
  El backend se implementó con arquitectura MVC simple, evitando complejidad innecesaria.
* **MD-02.2 – Refactorización continua:**  
  Rutas, controladores y modelos fueron mejorados constantemente para mantener consistencia y claridad.
* **MD-02.3 – Iteraciones cortas:**  
  Cada Avance académico (Avance 1, Avance 2, Avance 3) funcionó como un sprint XP enfocado en nuevas funciones.
* **MD-02.4 – Retroalimentación inmediata:**  
  Se realizaron pruebas en navegador y Postman después de cada cambio, detectando errores rápidamente.

**Beneficios obtenidos**

* Código ordenado, limpio y fácil de mantener.
* Incorporación constante de nuevas funciones sin romper la arquitectura previa.
* Mayor estabilidad del backend gracias a pruebas frecuentes.
* Desarrollo incremental y más seguro.

MD-03 – Justificación de la Metodología Híbrida Kanban + XP

**Nombre:** Metodología Híbrida Kanban + XP  
**Tipo:** Combinada (gestión visual + ingeniería ágil)  
**Aplicación:** Durante todo el ciclo de vida del proyecto  
**Propósito:** Equilibrar organización del flujo de trabajo con calidad técnica

**Descripción**

La combinación de Kanban y XP se seleccionó específicamente por su adecuación a un desarrollo individual y a los plazos académicos. Permite mantener claridad visual, avanzar de forma constante y asegurar calidad técnica.

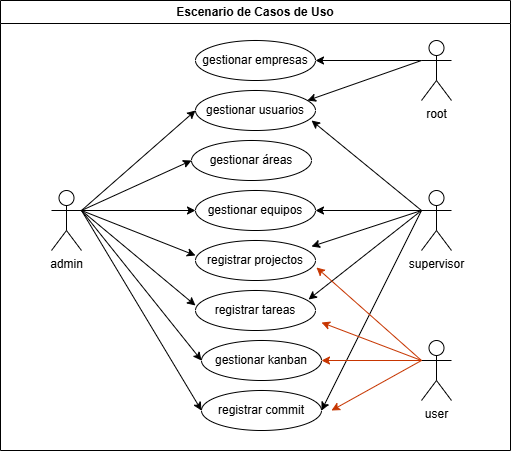
**Razones de la elección**

* **MD-03.1 – Control visual del trabajo:**  
  Kanban permitió ver de inmediato el estado del proyecto y evitar sobrecarga.
* **MD-03.2 – Calidad técnica continua:**  
  XP garantizó refactorización, simplicidad y pruebas permanentes.
* **MD-03.3 – No requiere reuniones formales:**  
  A diferencia de Scrum, esta combinación es más eficiente para proyectos individuales.
* **MD-03.4 – Alta adaptabilidad:**  
  Se ajusta fácilmente a entregas semanales y exigencias académicas.
* **MD-03.5 – Entrega continua:**  
  Favorece pequeños incrementos funcionales, ideales para el avance de TasksFlow.

**Resultados obtenidos**

* Flujo de trabajo ordenado y priorización efectiva.
* Desarrollo modular, incremental y escalable.
* Código backend más limpio, seguro y mantenible.
* Mayor trazabilidad del proceso de desarrollo.

## Casos de Uso del Sistema TasksFlow



CU-01 – Gestionar Empresas

**ID:** CU-01  
**Nombre:** Gestionar Empresas  
**Actor principal:** Root  
**Actores secundarios:** Ninguno  
**Objetivo:** Registrar, modificar o desactivar empresas para habilitar su uso dentro del sistema.

**Precondiciones**

* El actor debe haber iniciado sesión como usuario Root.
* El sistema debe estar operativo y conectado a la base de datos.

**Flujo Normal**

1. El Root accede al panel de administración global.
2. Selecciona la opción “Registrar Empresa”.
3. Ingresa nombre, descripción y estado inicial.
4. Asigna un usuario administrador para la empresa.
5. Confirma el registro.
6. El sistema crea un company\_id único y registra la empresa.
7. El sistema muestra la lista actualizada de empresas.

**Flujos Alternativos**

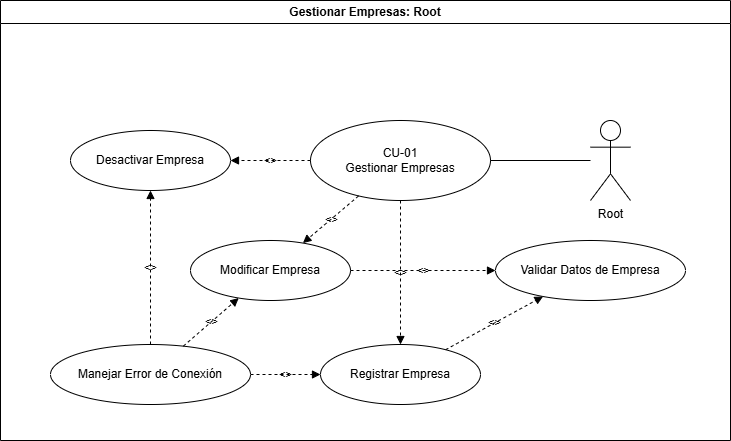
* **A1 – Datos incompletos**
  1. El sistema detecta campos vacíos o inválidos.
  2. Se muestra mensaje de error y solicita corregir.
* **A2 – Error de conexión**
  1. Si MySQL falla, el sistema muestra un mensaje y cancela la operación.

**Postcondiciones**

* La empresa queda registrada y disponible para asignación de usuarios.
* Se actualiza la lista de empresas visibles por el Root.

**Reglas de negocio**

* Solo el usuario Root puede registrar empresas.
* Ningún admin puede ver empresas ajenas.



CU-02 – Gestionar Usuarios

**ID:** CU-02  
**Nombre:** Gestionar Usuarios  
**Actor principal:** Admin / Root  
**Objetivo:** Crear, editar y desactivar usuarios dentro de una empresa.

**Precondiciones**

* El actor inició sesión.
* Root puede gestionar usuarios de todas las empresas.
* Admin solo puede gestionar usuarios de su propia empresa.

**Flujo Normal**

1. El actor accede al módulo “Usuarios”.
2. Selecciona “Crear Usuario”.
3. Ingresa nombre, correo, rol, área y estado.
4. Confirma el registro.
5. El sistema valida que el correo no esté duplicado.
6. El sistema encripta la contraseña (hash Argon2).
7. El sistema registra al usuario.
8. El sistema muestra la lista actualizada.

**Flujos Alternativos**

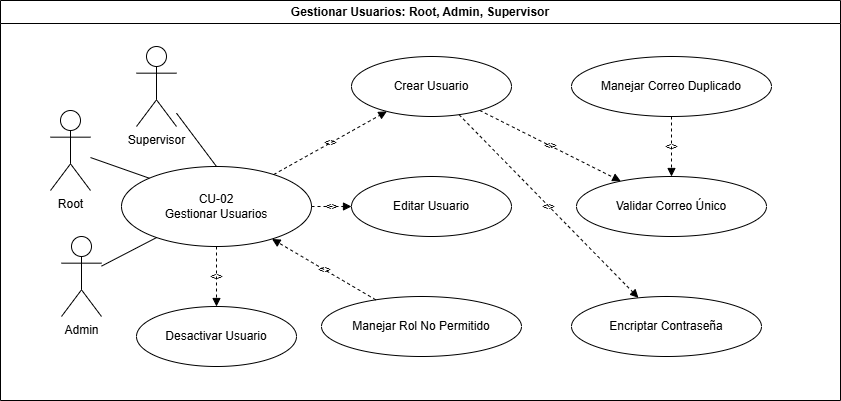
* **A1 – Correo duplicado**: El sistema alerta y solicita un correo diferente.
* **A2 – Rol no permitido**: El sistema restringe roles según permisos del actor.

**Postcondiciones**

* Usuario registrado y habilitado para iniciar sesión.
* Usuario aparece en los selectores de tareas, equipos y proyectos.

**Reglas de negocio**

* Un Admin no puede crear usuarios de otro company\_id.
* Los usuarios desactivados no pueden autenticarse.



CU-03 – Gestionar Áreas

**ID:** CU-03  
**Nombre:** Gestionar Áreas  
**Actor principal:** Admin  
**Objetivo:** Crear áreas funcionales dentro de la empresa.

**Precondiciones**

* El Admin debe haber iniciado sesión.
* La empresa debe existir.

**Flujo Normal**

1. Admin ingresa al módulo “Áreas”.
2. Selecciona “Nueva Área”.
3. Ingresa nombre, descripción y estado.
4. Guarda los cambios.
5. El sistema registra el área asociada al company\_id.
6. El sistema actualiza la lista de áreas.

**Flujos Alternativos**

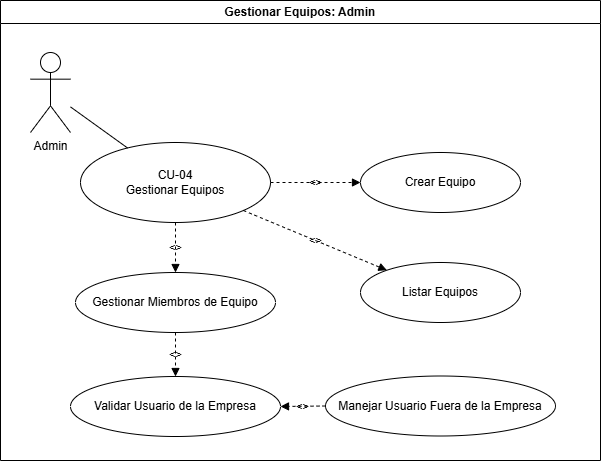
* Área duplicada → se muestra mensaje de advertencia.

**Postcondiciones**

* La nueva área queda disponible para asignar usuarios, equipos y proyectos.

**Reglas de negocio**

* El área no puede pertenecer a más de una empresa.



CU-04 – Gestionar Equipos de Trabajo

**ID:** CU-04  
**Nombre:** Gestionar Equipos  
**Actor principal:** Admin  
**Objetivo:** Crear y administrar equipos de trabajo internos.

**Precondiciones**

* El Admin está autenticado.
* Existen usuarios registrados en la empresa.

**Flujo Normal**

1. Admin abre el módulo “Equipos”.
2. Selecciona “Crear Equipo”.
3. Ingresa nombre, tipo, descripción y estado.
4. Agrega miembros al equipo.
5. Confirma creación.
6. El sistema registra el equipo y sus miembros.

**Flujos Alternativos**

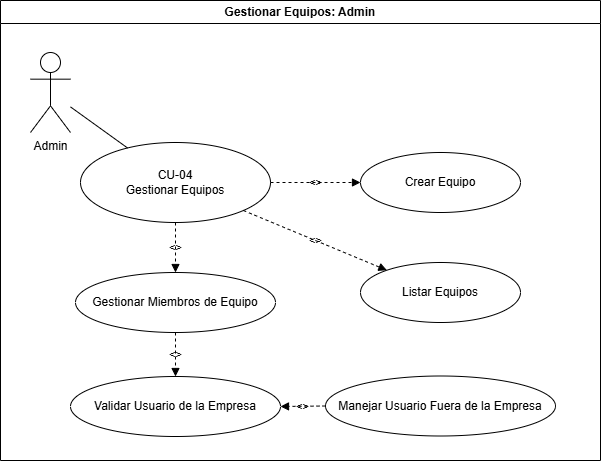
* Usuario no pertenece a la empresa → se muestra error.

**Postcondiciones**

* Equipo disponible para asignación de tareas y visualización en dashboards.

**Reglas de negocio**

* Un usuario solo puede pertenecer a equipos dentro de su empresa.



CU-05 – Registrar Proyectos

**ID:** CU-05  
**Nombre:** Registrar Proyectos  
**Actor principal:** Admin, Supervisor, User  
**Objetivo:** Crear proyectos para organizar tareas.

**Precondiciones**

* El actor debe estar autenticado.
* El actor debe tener permisos según rol:
  + Admin → toda la empresa
  + Supervisor → área asignada
  + User → proyectos personales

**Flujo Normal**

1. El actor abre el formulario “Nuevo Proyecto”.
2. Ingresa nombre, descripción, estado y fechas.
3. Selecciona área (si aplica según rol).
4. Confirma creación.
5. El sistema registra el proyecto con creator\_id y company\_id.
6. El sistema actualiza los selectores y dashboards.

**Flujos Alternativos**

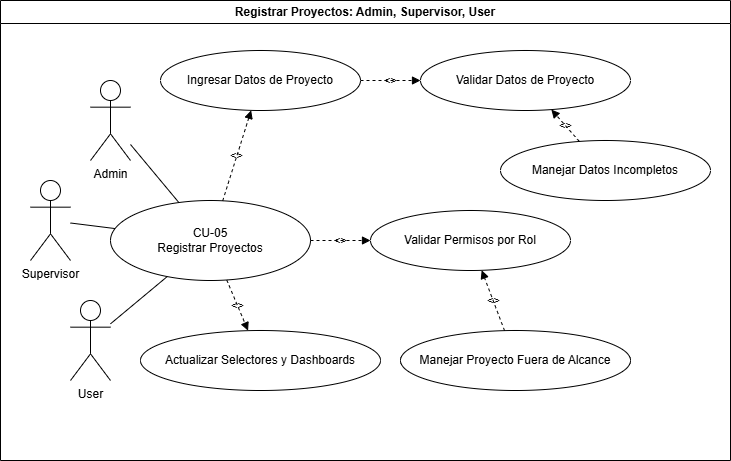
* Datos incompletos → el sistema solicita corrección.

**Postcondiciones**

* Proyecto disponible para tareas y filtros Kanban.

**Reglas de negocio**

* Un usuario no puede crear proyectos para otras áreas o empresas.



CU-06 – Registrar Tareas

**ID:** CU-06  
**Nombre:** Registrar Tareas  
**Actor principal:** Supervisor / Admin / User  
**Objetivo:** Crear tareas asignadas a usuarios o tareas personales.

**Precondiciones**

* El actor inició sesión.
* El actor tiene permisos sobre el área o equipo.
* Deben existir proyectos (opcional para tareas sueltas).

**Flujo Normal**

1. El actor selecciona “Nueva Tarea”.
2. Completa título, descripción, prioridad, estado inicial y fecha límite.
3. Selecciona usuarios asignados (excepto tareas personales).
4. Asocia un proyecto (opcional).
5. Confirma creación.
6. El sistema registra la tarea en tasks y las asignaciones en task\_assignments.

**Flujos Alternativos**

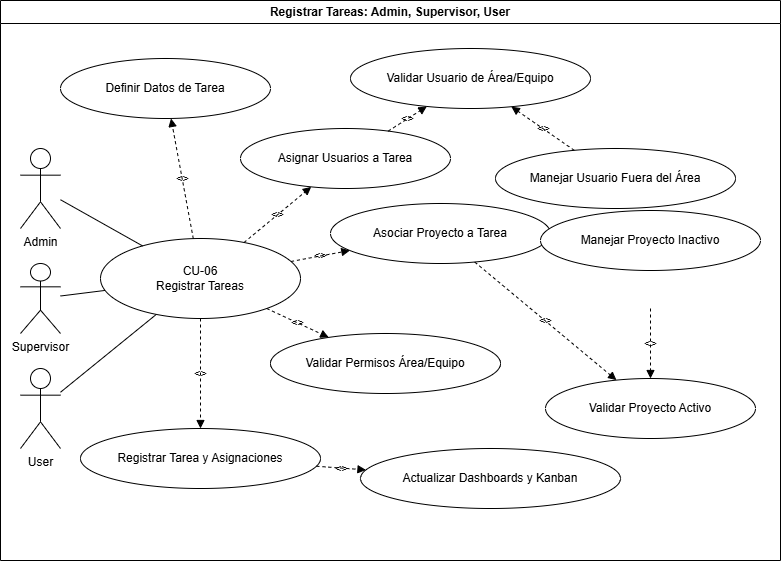
* Usuario asignado fuera del área → error.
* Proyecto inactivo → advertencia.

**Postcondiciones**

* La tarea queda disponible en dashboards y Kanban del rol correspondiente.

**Reglas de negocio**

* Las tareas personales solo pueden ser vistas por su creador.



CU-07 – Gestionar Tablero Kanban

**ID:** CU-07  
**Nombre:** Gestionar Kanban  
**Actor principal:** Admin / Supervisor / User  
**Objetivo:** Visualizar y actualizar tareas utilizando el tablero Kanban.

**Precondiciones**

* Existen tareas registradas.
* El usuario tiene acceso al módulo según su rol.

**Flujo Normal**

1. El actor activa la vista Kanban.
2. El sistema carga tareas clasificadas por estado.
3. El actor arrastra una tarea a otra columna.
4. El sistema actualiza el estado de la tarea.
5. El sistema refresca métricas y visualización.

**Flujos Alternativos**

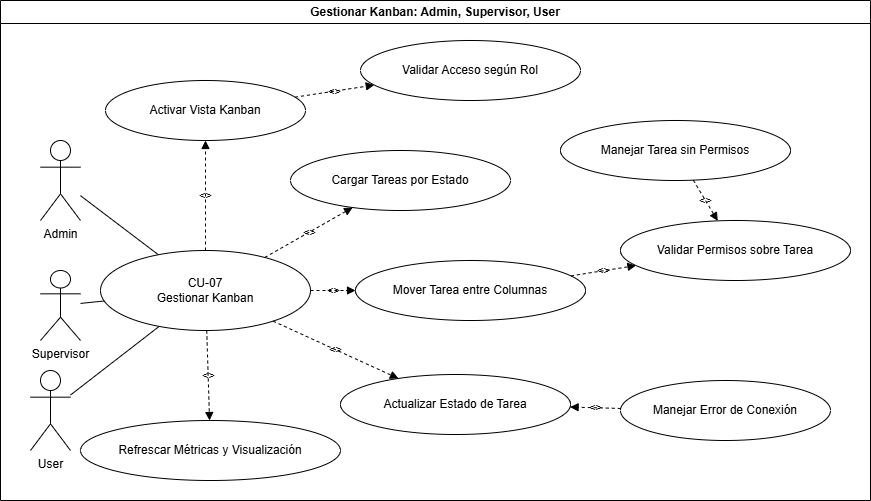
* Tarea sin permisos → movimiento bloqueado.
* Error de conexión → revertir cambio.

**Postcondiciones**

* Estado de la tarea actualizado en tiempo real.

**Reglas de negocio**

* Cada rol ve solo tareas permitidas (empresa/área/asignación).



CU-08 – Registrar Commit en una Tarea

**ID:** CU-08  
**Nombre:** Registrar Commit  
**Actor principal:** User / Supervisor  
**Objetivo:** Registrar mensajes y avances asociados a una tarea.

**Precondiciones**

* La tarea debe existir.
* El usuario debe estar asignado o ser supervisor.

**Flujo Normal**

1. El actor abre el modal de commits.
2. Escribe un mensaje descriptivo.
3. (Opcional) Cambia el estado de la tarea.
4. Confirma registro.
5. El sistema almacena el commit con autor, fecha y estado.
6. El sistema actualiza el historial visible.

**Flujos Alternativos**

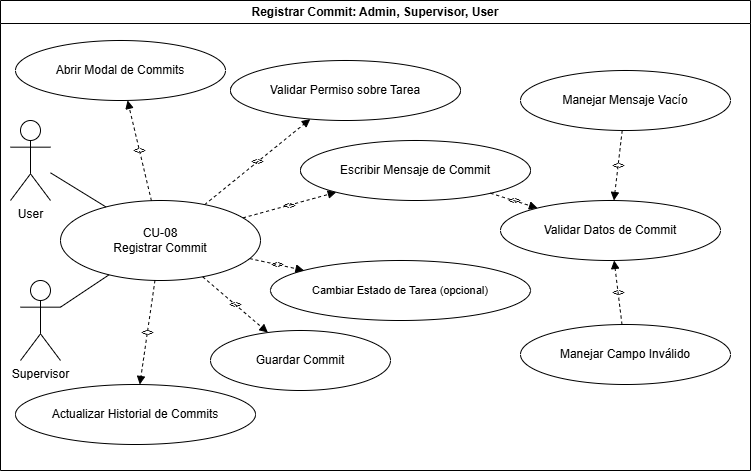
* Mensaje vacío → error.
* Campo inválido → corrección.

**Postcondiciones**

* El commit queda registrado en task\_commits.

**Reglas de negocio**

* Los commits no pueden eliminarse (trazabilidad obligatoria).



## Diagramas generales del sistema

Diagrama general estratégico del sistema (macroflujo TaskFlow)

**Objetivo:** mostrar la interacción global entre roles, proyectos, tareas, kanban y commits.

🔹 Actividades generales:

* Inicio de sesión según rol
* Acceso a vista correspondiente
* Operaciones del módulo (empresas, áreas, equipos, proyectos, tareas)
* Ciclo Kanban: crear → ejecutar → actualizar estado
* Registro de commits
* Finalización / cierre de actividad

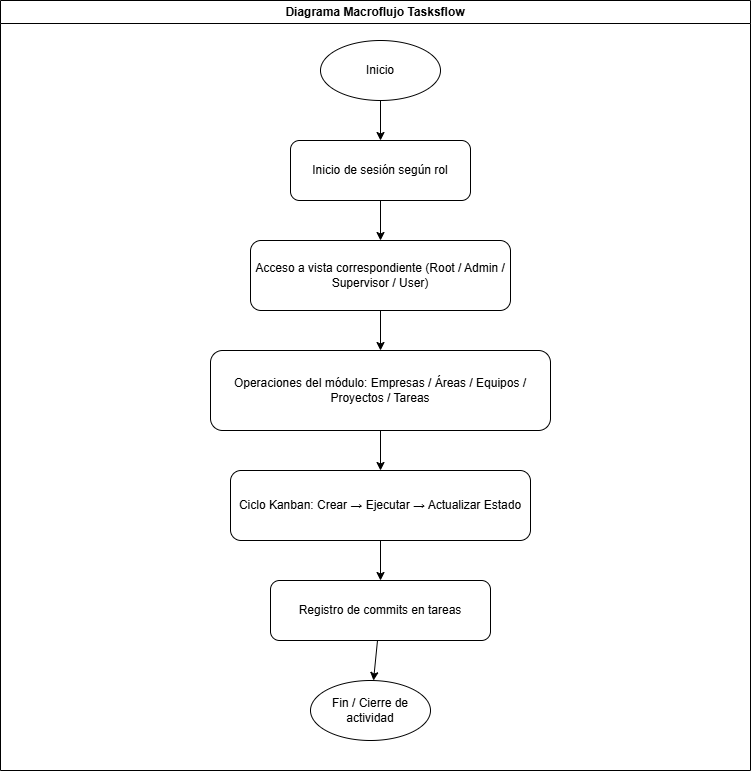


Diagrama de actividades del Admin (visión completa del rol)

**Admin gestiona toda la empresa:**

* Usuarios
* Áreas
* Equipos
* Proyectos
* Tareas si lo requiere
* Tablero Kanban general

🔹 Actividades:

* Abrir panel Admin
* Gestionar Empresa (si es root-admin)
* Gestionar áreas
* Gestionar equipos
* Gestionar usuarios
* Crear proyecto
* Asignar tareas
* Supervisar Kanban global
* Validar estados
* Cerrar sesión

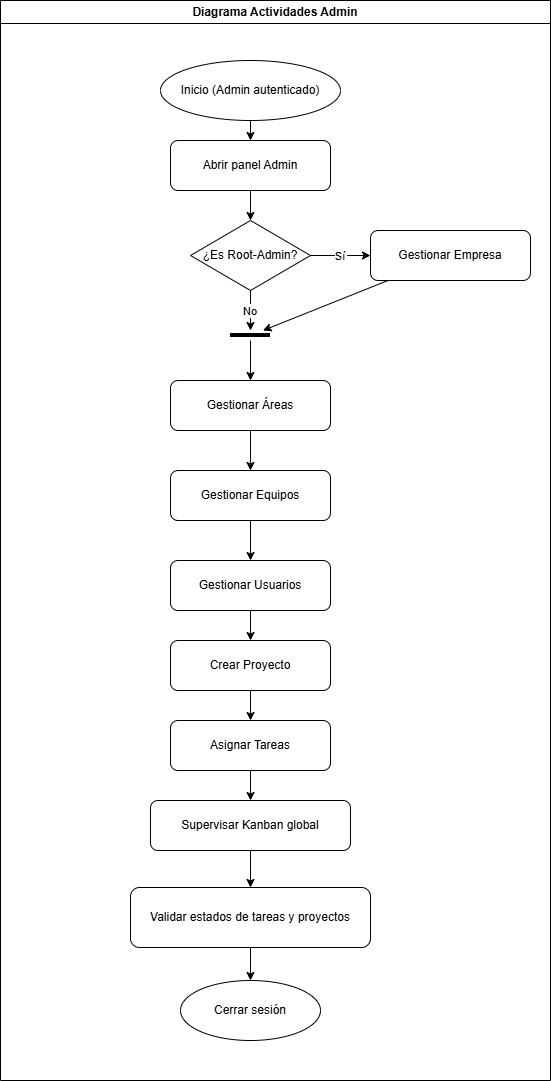


Diagrama de actividades del Supervisor (flujo táctico de coordinación)

El supervisor es el actor más importante a nivel medio:

🔹 Actividades:

* Ingresar al panel Supervisor
* Ver KPIs del equipo
* Ver área
* Crear tareas
* Actualizar tareas
* Mover estado en Kanban
* Revisar commits
* Validar avance del equipo
* Devolver tarea para revisión
* Marcar tarea completa
* Cerrar sesión

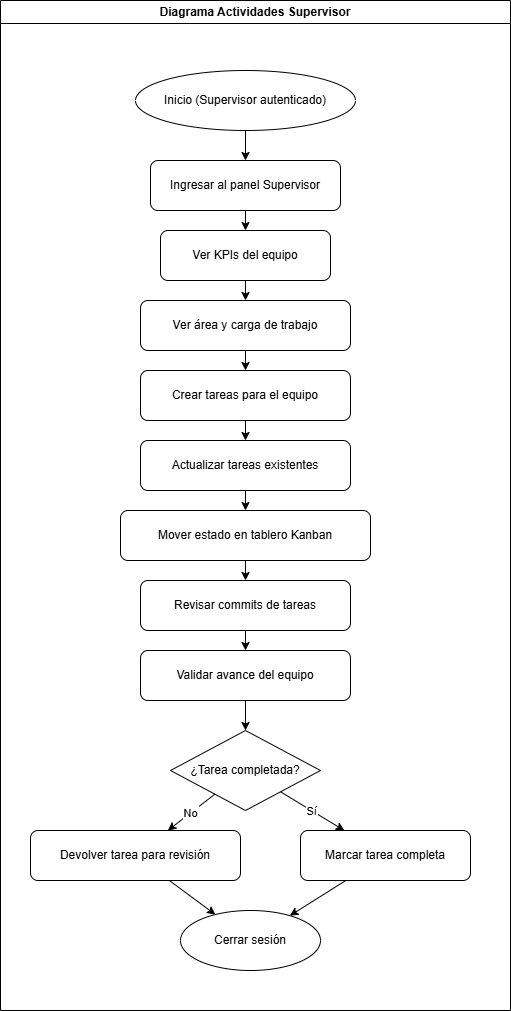
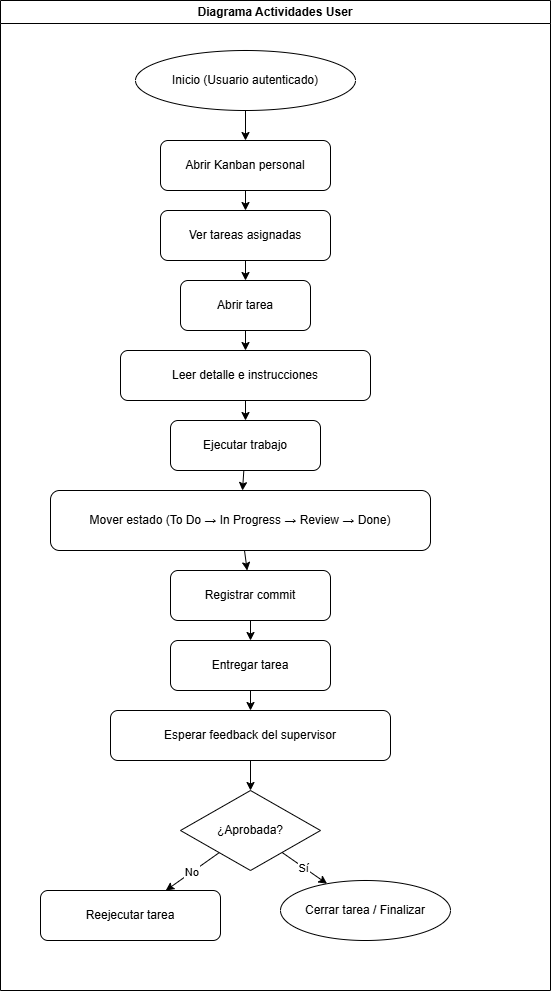


Diagrama de actividades del Usuario (flujo operativo Kanban)

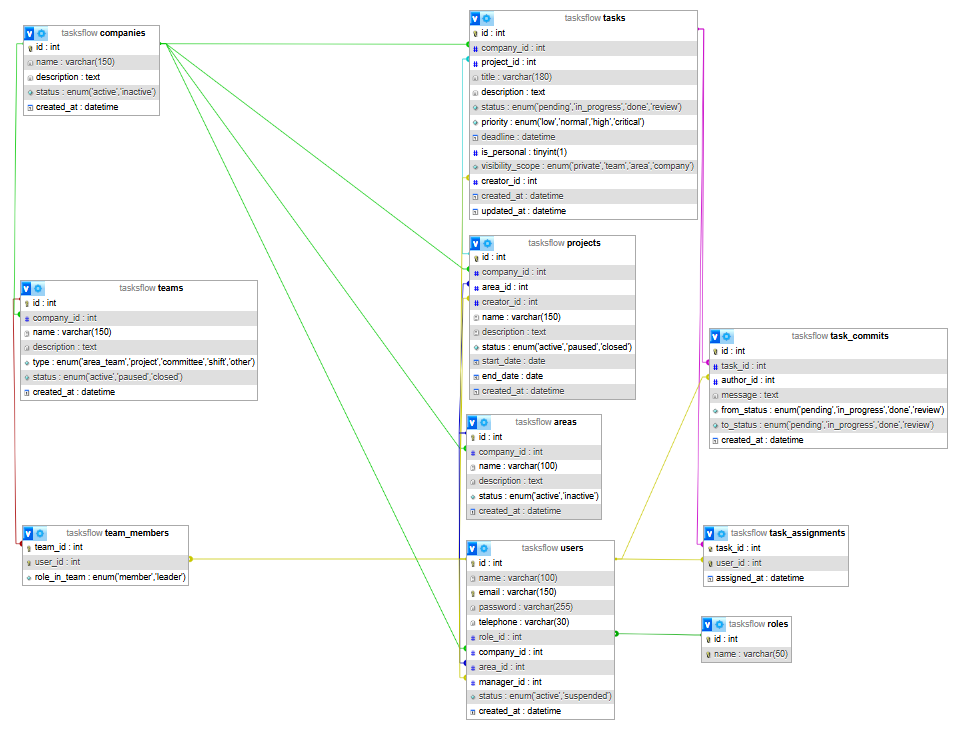
El usuario es quien ejecuta las tareas:

🔹 Actividades:

* Abrir Kanban personal
* Ver tareas asignadas
* Abrir tarea
* Leer detalle e instrucciones
* Ejecutar trabajo
* Mover estado (To Do → In Progress → Review → Done)
* Registrar commit
* Entregar tarea
* Esperar feedback del supervisor
* Reejecutar o cerrar



## MER Modelo Entidad Relación



## Diseño físico de la base de datos

Visión técnica (tablas principales y relaciones)

El sistema TasksFlow se implementa sobre una base de datos MySQL llamada **tasksflow**, creada con **codificación utf8mb4** para soportar correctamente caracteres especiales y acentos. El diseño físico se compone de un conjunto de tablas normalizadas que separan **empresa**, **estructura organizacional**, **usuarios**, **proyectos**, **tareas** y **trazabilidad**, manteniendo integridad referencial mediante claves foráneas y restricciones ON DELETE en los puntos críticos.

A continuación, se describen las **tablas núcleo** y sus relaciones principales.

1. Empresas y estructura organizacional

* **companies**
  + **Rol:** tabla raíz que representa a cada empresa cliente que utiliza TasksFlow.
  + **Campos clave:**
    - id INT PK, autoincremental.
    - name VARCHAR(150), description TEXT.
    - status ENUM('active','inactive').
    - created\_at DATETIME (timestamp de creación).
  + **Relaciones:**
    - 1:N con areas, users, teams, projects y tasks.
* **roles**
  + **Rol:** catálogo de roles del sistema (root, admin, supervisor, user).
  + **Campos:** id INT PK, name VARCHAR(50) UNIQUE.
  + **Uso:** permite desacoplar la lógica de permisos del resto del modelo.
* **areas**
  + **Rol:** representa áreas funcionales dentro de cada empresa (por ejemplo, Operaciones, TI, RRHH).
  + **Campos clave:**
    - id INT PK.
    - company\_id INT FK → companies.id.
    - name, description, status ENUM('active','inactive'), created\_at.
  + **Relaciones:**
    - 1:N con users (usuarios asignados al área).
    - 1:N con projects (proyectos del área).

2. Usuarios y roles

* **users**
  + **Rol:** tabla núcleo de personas que utilizan TaskFlow.
  + **Campos principales:**
    - id INT PK.
    - name, email UNIQUE, password (hash Argon2), telephone.
    - role\_id INT FK → roles.id.
    - company\_id INT FK → companies.id.
    - area\_id INT FK → areas.id.
    - manager\_id INT FK auto–referenciada → users.id (jefatura o supervisor).
    - status ENUM('active','suspended'), created\_at.
  + **Relaciones:**
    - 1:N como **creador** en projects.creator\_id y tasks.creator\_id.
    - 1:N como **autor** en task\_commits.author\_id.
    - N:M con tasks a través de task\_assignments.
    - 1:N con team\_members (pertenencia a equipos).

3. Equipos de trabajo

* **teams**
  + **Rol:** modela equipos o grupos de trabajo dentro de una empresa (cuadrillas, comités, turnos, etc.).
  + **Campos:**
    - id INT PK.
    - company\_id INT FK → companies.id.
    - name, description.
    - type ENUM('area\_team','project','committee','shift','other').
    - status ENUM('active','paused','closed').
    - created\_at DATETIME.
* **team\_members**
  + **Rol:** resuelve la relación **N:M entre users y teams**.
  + **Campos:**
    - team\_id INT FK → teams.id.
    - user\_id INT FK → users.id.
    - role\_in\_team ENUM('member','leader').
    - PK compuesta (team\_id,user\_id).
  + **Restricciones:**
    - ON DELETE CASCADE en ambos FKs para eliminar membresías cuando se borra un equipo o usuario.

Este diseño permite que un usuario pertenezca a múltiples equipos dentro de la misma empresa, manteniendo además el rol que desempeña en cada uno.

4. Proyectos

* **projects**
  + **Rol:** agrupa y organiza tareas en unidades de trabajo de mediano/largo plazo (proyectos operacionales).
  + **Campos principales:**
    - id INT PK.
    - company\_id INT FK → companies.id.
    - area\_id INT FK → areas.id (área responsable).
    - creator\_id INT FK → users.id (quien crea el proyecto).
    - name, description.
    - status ENUM('active','paused','closed').
    - start\_date, end\_date, created\_at.
  + **Relaciones:**
    - 1:N con tasks (tareas asociadas al proyecto).

5. Tareas y asignaciones

* **tasks**
  + **Rol:** tabla central del flujo Kanban, representa cada tarea operativa.
  + **Campos principales:**
    - id INT PK.
    - company\_id INT FK → companies.id.
    - project\_id INT FK → projects.id (puede ser NULL para tareas sueltas).
    - title VARCHAR(180), description TEXT.
    - status ENUM('pending','in\_progress','done','review') con valor por defecto 'pending'.
    - priority ENUM('low','normal','high','critical').
    - deadline DATETIME (fecha límite).
    - is\_personal BOOLEAN (tareas personales del usuario).
    - visibility\_scope ENUM('private','team','area','company') para controlar el alcance de visualización.
    - creator\_id INT FK → users.id.
    - created\_at, updated\_at.
  + **Relaciones:**
    - 1:N con task\_commits (historial de cambios de la tarea).
    - N:M con users a través de task\_assignments.
* **task\_assignments**
  + **Rol:** modela la relación **N:M entre usuarios y tareas**, permitiendo múltiples responsables por tarea.
  + **Campos:**
    - task\_id INT FK → tasks.id.
    - user\_id INT FK → users.id.
    - assigned\_at DATETIME.
    - PK compuesta (task\_id,user\_id).
  + **Restricciones:**
    - ON DELETE CASCADE desde tasks y users, garantizando que no queden asignaciones huérfanas.

6. Trazabilidad tipo “Git” de las tareas

* **task\_commits**
  + **Rol:** implementa la trazabilidad fina de tareas, similar a un historial de commits de Git.
  + **Campos principales:**
    - id INT PK autoincremental.
    - task\_id INT FK → tasks.id.
    - author\_id INT FK → users.id.
    - message TEXT (descripción del cambio).
    - from\_status y to\_status ENUM('pending','in\_progress','done','review') para registrar transición de estado.
    - created\_at DATETIME.
  + **Uso:**
    - Permite reconstruir qué cambios se hicieron, por quién y cuándo, mejorando la auditoría y el control de avance.

7. Consideraciones de implementación física

* **Integridad referencial:**  
  Todas las tablas claves (users, tasks, projects, companies, etc.) se relacionan mediante claves foráneas explícitas, lo que evita registros huérfanos y asegura consistencia de la información.
* **Índices y claves primarias:**
  + Cada tabla tiene una **PK autoincremental** (id) o una PK compuesta en tablas asociativas (team\_members, task\_assignments).
  + Las FKs generan índices implícitos que optimizan las consultas por empresa, área, proyecto o usuario.
* **Tipos ENUM y BOOLEAN:**  
  Se utilizan tipos ENUM para campos de estado, prioridad, tipo de equipo y visibilidad, reduciendo errores de integridad semántica (solo valores válidos) y simplificando validaciones a nivel de aplicación. El uso de BOOLEAN/TINYINT(1) en flags como is\_personal hace más eficiente el almacenamiento.
* **Escalabilidad lógica:**
  + La separación entre tasks, task\_assignments y task\_commits permite escalar el número de usuarios y el volumen de historial sin modificar el modelo base.
  + La normalización en tablas companies, areas, teams y roles hace posible soportar múltiples empresas clientes en un mismo esquema, manteniendo aislamiento lógico por company\_id.

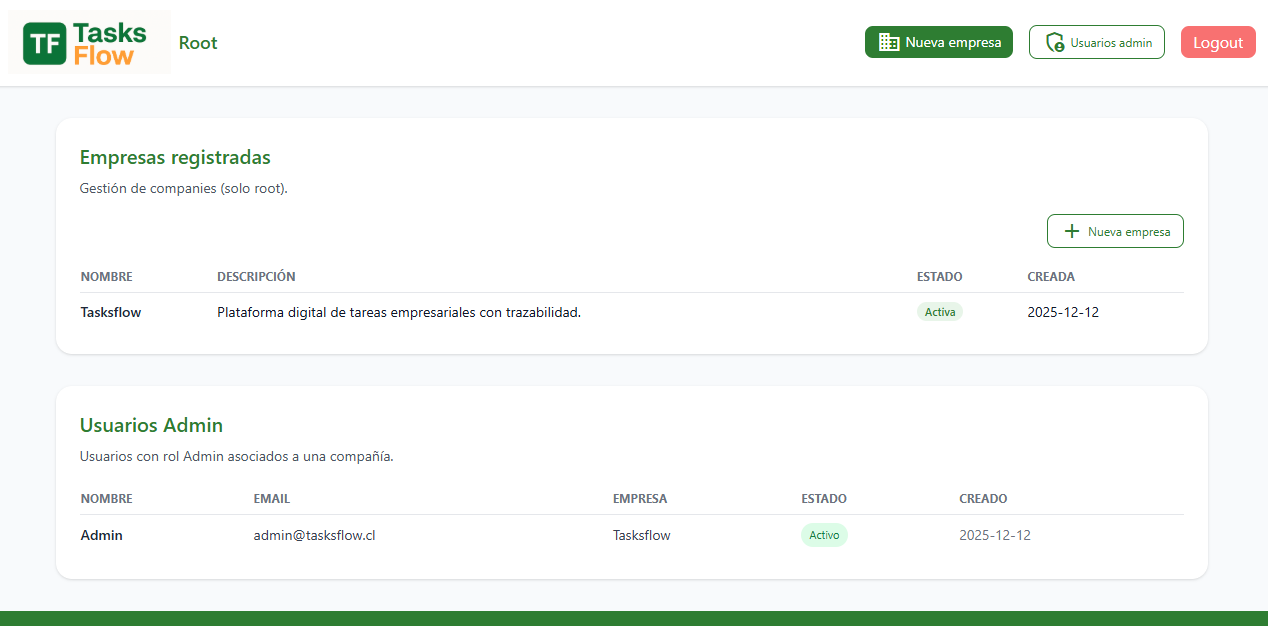
## Wireframes del Sistema

Introducción a los Wireframes

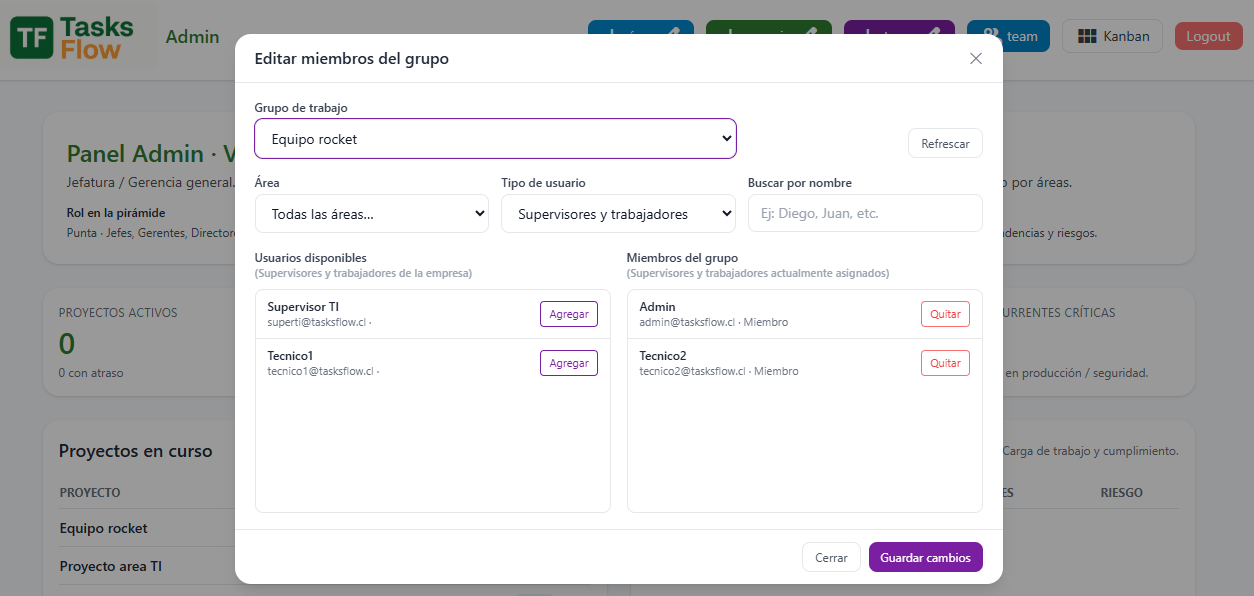
Los wireframes representan la estructura visual preliminar de las interfaces del sistema **TasksFlow**, permitiendo definir la organización de elementos, distribución funcional y navegabilidad antes del desarrollo final.  
Su objetivo es mostrar, de manera simple y clara, cómo interactúan los distintos roles dentro de la plataforma y qué componentes funcionales están disponibles para cada uno.

En esta etapa se establecen:

* La **jerarquía visual** de cada vista.
* Los **componentes clave** de interacción (botones, menús, paneles, tableros).
* La **distribución funcional** según el rol del usuario (Root, Admin, Supervisor, User).
* La **coherencia entre las pantallas**, preparando el flujo para la fase de Wireflow.

1: Vista Root – Administración del Sitio

**Descripción breve:**  
La vista Root permite la gestión global del sistema, incluyendo administración de empresas registradas, configuración inicial del entorno, y privilegios de nivel superior.

2. Vista Team Admin — Gestión interna de equipos

**Descripción breve:**

La vista *Team Admin* corresponde al módulo de gestión de equipos dentro del panel administrativo del sistema. Desde este entorno, el administrador puede **crear, organizar y administrar equipos de trabajo**, definiendo su nombre, descripción y estado, así como su vinculación con la estructura interna de la empresa.

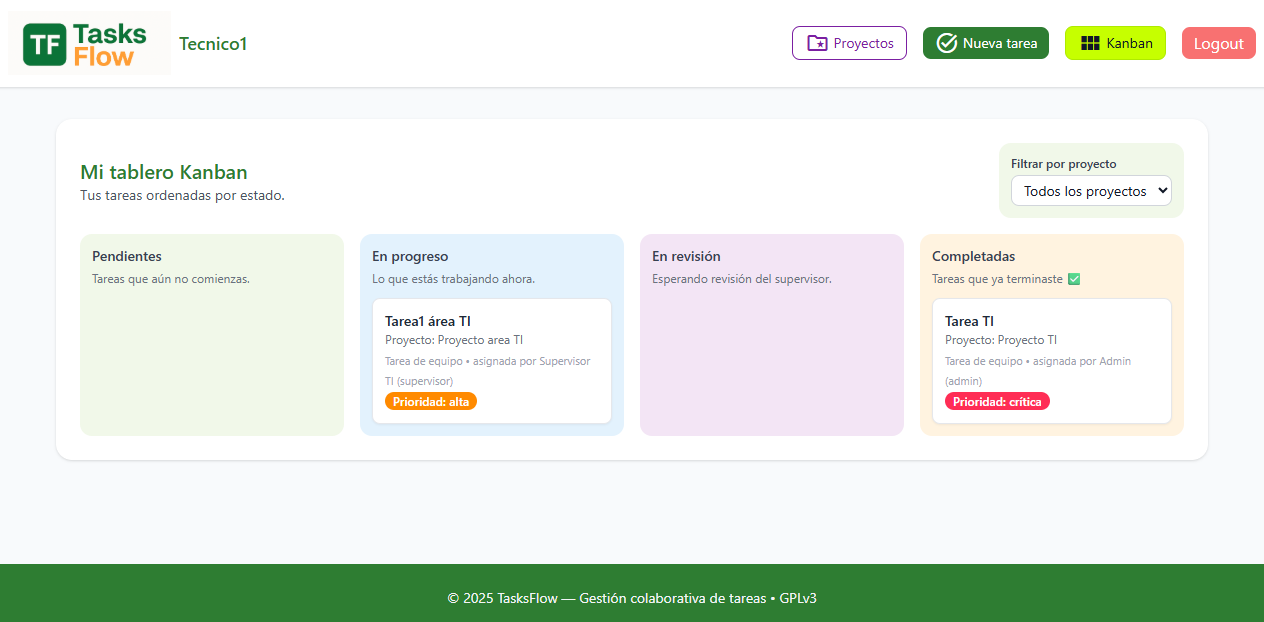
3: **Vista Dashboard Supervisor — Supervisión táctica de equipos**



**Descripción breve:**

El *Dashboard del Supervisor* presenta una visión táctica orientada a la coordinación diaria del equipo de trabajo, permitiendo monitorear el estado de las tareas, la carga laboral por integrante y el nivel de cumplimiento de objetivos en el corto plazo.

4: Vista Kanban — Usuario



**Descripción breve:**  
El Kanban del usuario final presenta únicamente las tareas asignadas a él, con una interfaz centrada en la ejecución operativa.  
Incluye funciones como: abrir tareas, registrar commits, arrastrar entre columnas y finalizar actividades.  
El foco está en la simplicidad y en la claridad del estado de cada tarea.

## Wireflows del Sistema

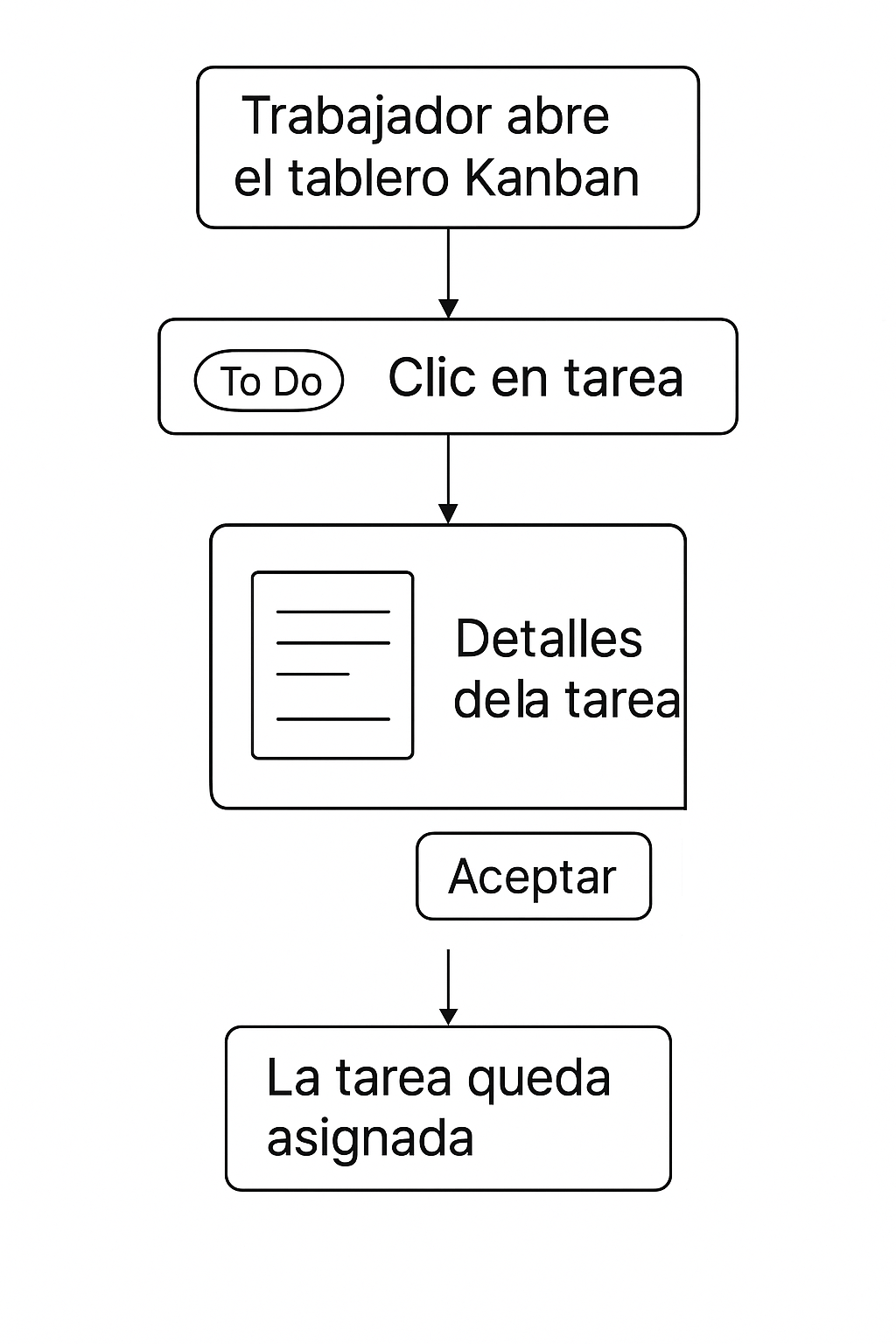
Introducción a los Wireflows

Los wireflows representan la **secuencia de interacción del usuario con el sistema**, combinando la estructura visual definida en los wireframes con el **flujo de acciones y decisiones** que realizan los distintos roles dentro de la plataforma TasksFlow.

Su objetivo es describir, de manera clara y ordenada, **cómo se ejecutan los procesos funcionales principales**, desde la acción inicial del usuario hasta el resultado final en el sistema. A diferencia de los wireframes, los wireflows ponen énfasis en el **comportamiento del sistema**, la navegación entre estados y la respuesta ante las acciones del usuario.

En esta etapa se definen:

* Los pasos secuenciales de interacción por rol.
* Las acciones disponibles en cada vista.
* Los cambios de estado de las tareas dentro del tablero Kanban.
* La relación entre usuarios, tareas y notificaciones.
* La coherencia entre las vistas operativas y tácticas del sistema.

****Wireflow 1: Usuario / Trabajador – Visualización y asignación de tarea

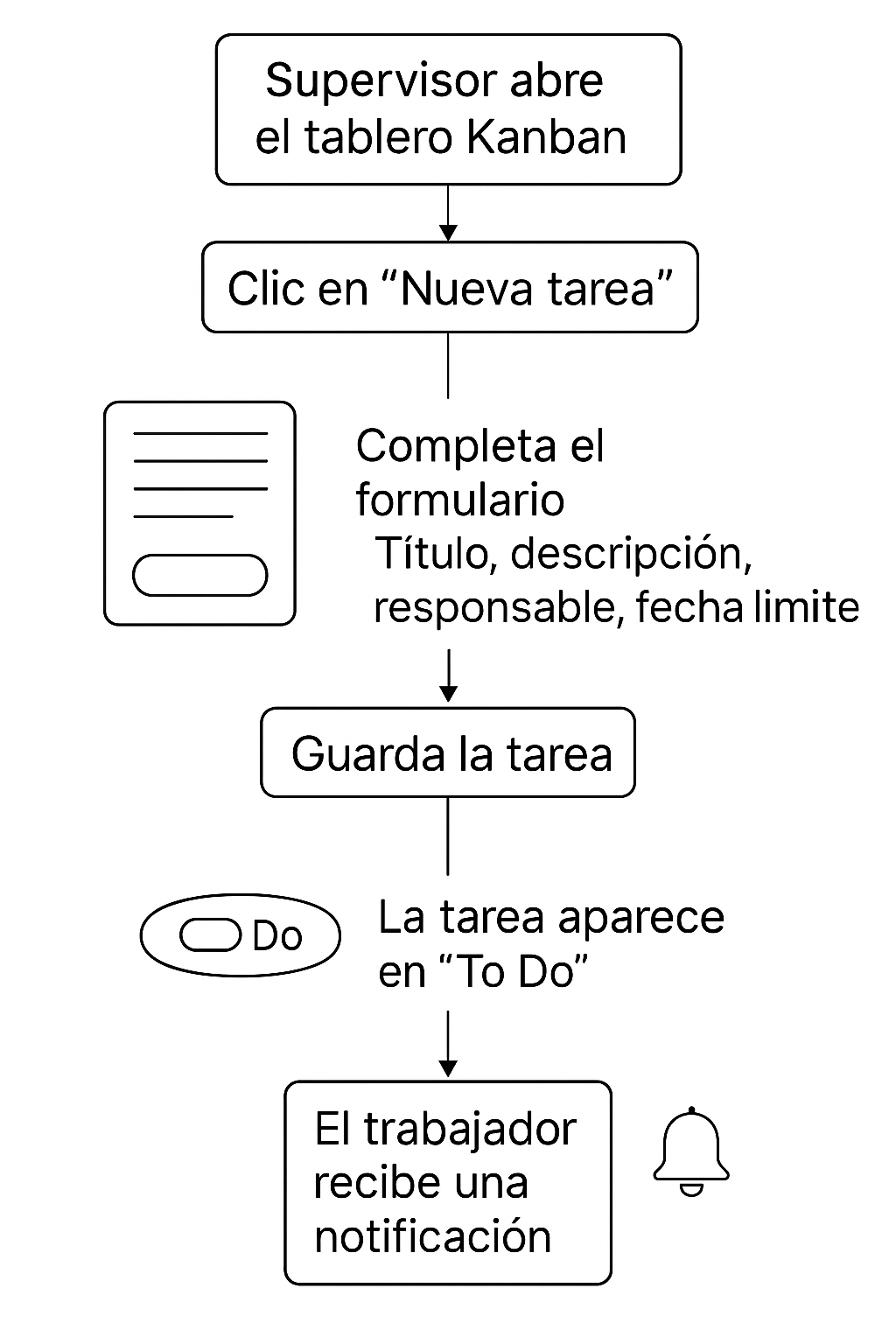
**Descripción del flujo:**

Este wireflow describe el proceso mediante el cual un **trabajador** interactúa con el tablero Kanban para revisar y aceptar una tarea asignada.

El flujo comienza cuando el trabajador accede al tablero Kanban y visualiza las tareas disponibles en la columna *To Do*. Al seleccionar una tarea, el sistema despliega un panel de **detalles de la tarea**, donde se presenta información relevante como descripción, responsable y estado.

Una vez revisada la información, el trabajador confirma la acción mediante el botón *Aceptar*, quedando la tarea formalmente asignada y registrada dentro de su flujo de trabajo. Este proceso garantiza claridad en la asignación y trazabilidad de la responsabilidad sobre la tarea.

Wireflow 2: Supervisor – Creación y asignación de nueva tarea

****

**Descripción del flujo:**

Este wireflow representa el proceso mediante el cual el **Supervisor** crea y asigna una nueva tarea a un miembro de su equipo desde el tablero Kanban.

El flujo inicia cuando el supervisor accede al tablero Kanban y selecciona la opción *Nueva tarea*. A continuación, el sistema presenta un formulario donde el supervisor ingresa los datos esenciales de la tarea, tales como:

* Título,
* Descripción,
* Responsable,
* Fecha límite.

Una vez completado el formulario, el supervisor guarda la tarea, la cual es incorporada automáticamente en la columna *To Do* del tablero. Como resultado, el trabajador asignado recibe una **notificación**, asegurando una comunicación inmediata y una correcta distribución de las responsabilidades.

Este flujo refuerza el rol táctico del supervisor, facilitando la planificación operativa, la asignación eficiente de recursos y el control del avance del equipo.

## Plan de Pruebas

**Objetivo**

Validar que el sistema cumple con los requerimientos funcionales y no funcionales definidos, asegurando estabilidad, seguridad y coherencia en la operación de TasksFlow.

**Alcance**

Las pruebas se enfocan en:

* Autenticación y control de acceso.
* Gestión de proyectos y tareas.
* Tablero Kanban.
* Registro de commits y trazabilidad.

**Casos de prueba**

| **ID** | **Requerimiento** | **Entrada** | **Resultado esperado** | **Estado** |
| --- | --- | --- | --- | --- |
| CP-01 | RF-10 | Email y contraseña válidos | Acceso según rol | OK |
| CP-02 | RF-05 | Datos válidos de proyecto | Proyecto creado y visible | OK |
| CP-03 | RF-06 | Datos de tarea con asignados | Tarea creada en estado pending | OK |
| CP-04 | RF-08 | Arrastre Kanban | Cambio de estado persistente | OK |
| CP-05 | RF-09 | Commit con mensaje | Commit registrado y visible | OK |
| CP-06 | RNF-02 | Usuario sin permisos | Acceso denegado | OK |

**Criterio de aceptación**

* El resultado obtenido coincide con el esperado.
* Existe evidencia (captura o log).
* No genera errores colaterales.

## Manual de Usuario

**Acceso al sistema**

El usuario accede mediante correo electrónico y contraseña. El sistema identifica automáticamente el rol y carga la vista correspondiente.

**Rol Root – Administrador Global**

Funciones principales:

* Registrar, editar y desactivar empresas.
* Asignar administradores por empresa.
* Supervisar el estado global del sistema.

Flujo básico:

1. Iniciar sesión como Root.
2. Acceder al panel de empresas.
3. Crear o modificar empresas y administradores.

**Rol Admin – Administración de Empresa**

Funciones principales:

* Gestionar usuarios, áreas y equipos.
* Crear proyectos y tareas.
* Visualizar Kanban a nivel empresa.

Flujo básico:

1. Acceder al dashboard Admin.
2. Configurar estructura interna (áreas, equipos).
3. Crear proyectos y asignar tareas.
4. Supervisar avance en Kanban.

**Rol Supervisor – Coordinación de Área**

Funciones principales:

* Crear y asignar tareas a su equipo.
* Visualizar métricas del área.
* Revisar commits y validar tareas.

Flujo básico:

1. Ingresar al panel Supervisor.
2. Crear tareas para su área.
3. Controlar el avance mediante Kanban.
4. Revisar historial de commits.

**Rol User – Ejecutor de Tareas**

Funciones principales:

* Ver tareas asignadas y personales.
* Mover tareas en el Kanban.
* Registrar commits de avance.

Flujo básico:

1. Acceder al Kanban personal.
2. Ejecutar tareas asignadas.
3. Registrar avances mediante commits.
4. Finalizar tareas.

## Conclusiones

El desarrollo del proyecto **TasksFlow – Gestión de Tareas Empresariales con Trazabilidad** permitió abordar de manera integral un problema frecuente en las organizaciones modernas: la falta de una herramienta centralizada, clara y trazable para la gestión de tareas y actividades diarias. A lo largo del Taller de Proyecto de la Especialidad, se logró pasar desde el análisis del problema hasta la implementación de un prototipo funcional, cumpliendo con los objetivos académicos y técnicos propuestos inicialmente.

Desde el punto de vista funcional, el sistema implementado satisface los requerimientos definidos, permitiendo la gestión de empresas, usuarios, áreas, equipos, proyectos y tareas, con un control de acceso basado en roles y una clara segmentación por empresa. El uso de un tablero Kanban como eje visual facilita la comprensión del estado del trabajo y mejora la coordinación entre los distintos niveles organizacionales: administrativo, táctico y operativo.

Uno de los principales aportes del proyecto es la incorporación de un **historial de acciones tipo “Git” aplicado a la gestión de tareas**, lo que introduce un nivel de trazabilidad poco habitual en sistemas de este tipo. Esta característica permite registrar decisiones, avances y cambios de estado, fortaleciendo la transparencia, la rendición de cuentas y el seguimiento histórico de cada tarea, aspectos clave en entornos empresariales.

En el ámbito técnico, TasksFlow fue desarrollado utilizando tecnologías actuales y ampliamente utilizadas en la industria, como Node.js, Express y MySQL, organizadas bajo una arquitectura MVC que favorece la mantenibilidad y escalabilidad del sistema. La aplicación de prácticas ágiles, combinando Kanban personal y principios de Extreme Programming, permitió avanzar de forma incremental, mantener calidad técnica y adaptarse a los plazos académicos del proyecto.

Asimismo, la elaboración de diagramas, modelos de datos, wireframes, wireflows, plan de pruebas y manual de usuario contribuyó a una documentación completa y coherente, demostrando la importancia de no limitar el desarrollo de software únicamente a la programación, sino de considerar el ciclo completo del sistema, desde el análisis hasta la validación y uso final.

Finalmente, se concluye que TasksFlow constituye un **prototipo funcional sólido**, alineado con necesidades reales del entorno empresarial y con potencial de evolución. Como trabajo futuro, el sistema podría ampliarse incorporando notificaciones automáticas, reportes avanzados, métricas de desempeño, integración con servicios externos y mejoras en la experiencia de usuario. No obstante, el proyecto cumple plenamente con los objetivos planteados y representa una experiencia formativa significativa en el proceso de formación profesional del autor.