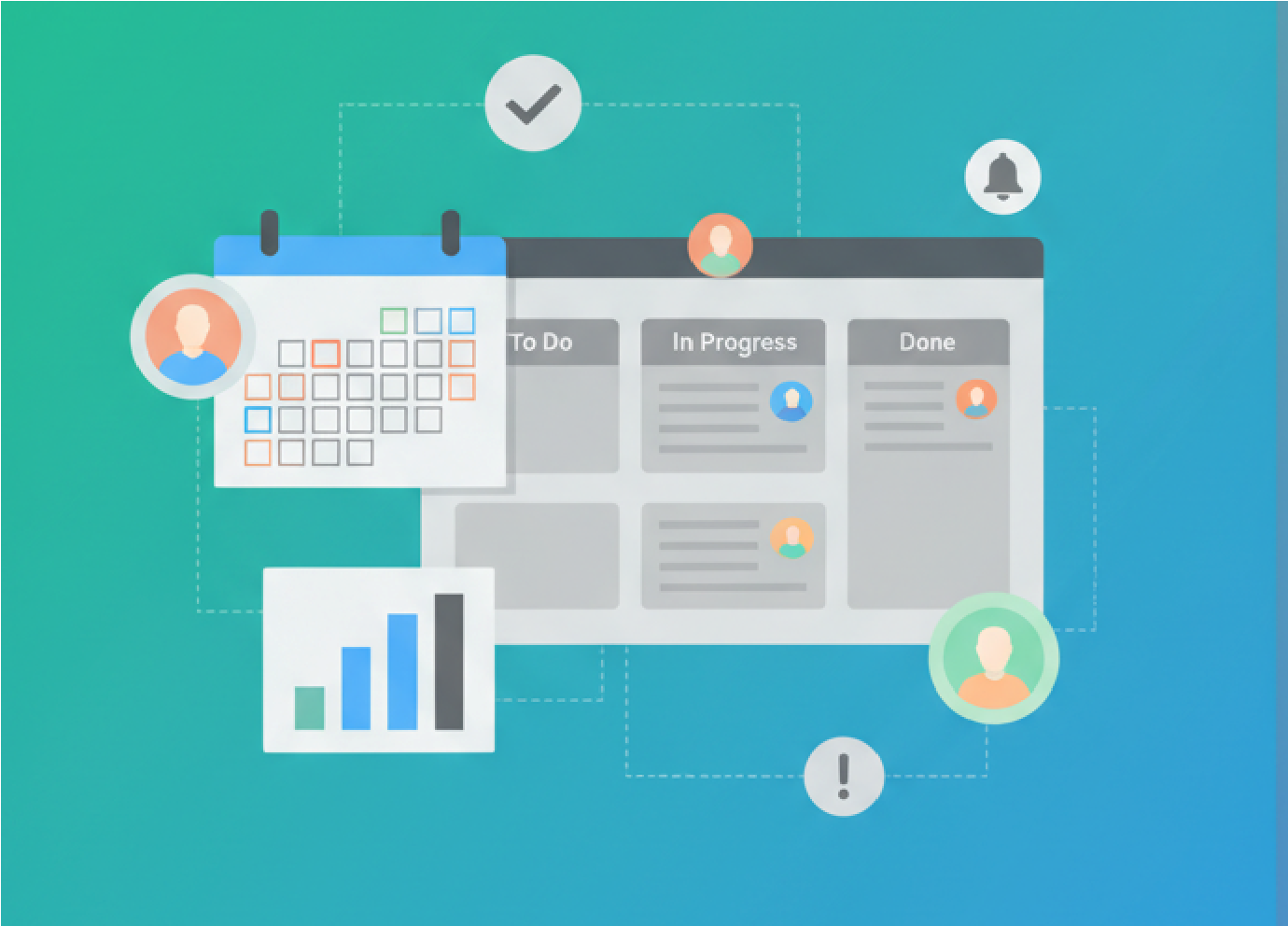
**Diseño de Software I, II y III**

**Creación de módulo de gestión de tareas (Task Assistant)**

**Edwin José Bertel Negrete:** Ebertelnegrete38@correo.unicordoba.edu.co

**Álvaro Pio Villalba Páez**: Avillalbapaez52@correo.unicordoba.edu.co

**Jesi José Correa Galván:** Jcorreagalvan76@correo.unicordoba.edu.co



# Descripción del software

Task Assistant es una plataforma web de gestión de tareas educativas que centraliza la planificación, asignación y seguimiento de actividades académicas, reduciendo la carga administrativa del docente y facilitando la ejecución y el autocontrol del estudiante en un entorno único, trazable y colaborativo; ofrece gestión integral de tareas (crear, editar, duplicar, eliminar y recuperar desde papelera), asignación inteligente manual o automatizada con notificaciones inmediatas, colaboración contextual mediante comentarios anidados y menciones, recursos educativos con adjuntos y enlaces (control básico de versiones), seguimiento en tiempo real con estado, porcentaje de avance, registro de tiempo y métricas, organización flexible por proyectos con etiquetas (color y palabra), anclado, búsqueda y filtros, planificación temporal mediante metas semanales y mensuales y sincronización con Google Calendar/Outlook, evaluación con calificación y retroalimentación, reportes con exportación a PDF/Excel, y seguridad mediante control de acceso por roles (Administrador, Docente, Estudiante) e historial de cambios para auditoría y trazabilidad.

**Etapa 1 Diseño de la Aplicación y Análisis de Requisitos**

**1.Introducción**

# Propósito del Documento

Este documento define las especificaciones técnicas y funcionales de Task Assistant, una plataforma web de gestión de tareas educativas orientada a contextos universitarios y escolares. Sirve como guía de diseño e implementación para el equipo de desarrollo, como medio de comunicación del alcance para los actores académicos y como base para la validación y evaluación del proyecto en la asignatura de Diseño de Software. Incluye los casos de uso priorizados (25), requisitos no funcionales (desempeño, seguridad, usabilidad, escalabilidad), modelado de datos (E/R, relacional y script SQL) y anexos (UML y wireframes).

**Etapas del proyecto**

# Etapa 1 Diseño de la Aplicación y Análisis de Requisitos

Se caracterizó y acotó Task Assistant como un sistema de gestión de tareas educativas, definiendo con claridad su propósito y alcance. Como resultado de esta fase, se elaboró la descripción del sistema y de sus objetivos funcionales, operacionales y de experiencia de usuario; se documentaron casos de uso estandarizados en formato MoSCoW, incluyendo precondiciones, postcondiciones, flujos principales y alternos, excepciones, reglas de negocio, requerimientos no funcionales asociados y criterios de aceptación; se construyó una matriz de permisos por rol (Administrador, Docente Principal, Docente Invitado y Estudiante), señalando el alcance mediante la codificación ✓ / ✓\* / —; se diseñó el modelo de datos, integrando el diagrama entidad–relación, el diagrama relacional normalizado a 3FN, la definición de claves primarias y foráneas, así como los índices correspondientes, junto con el script SQL base para PostgreSQL 14 o superior; y, finalmente, se establecieron los lineamientos de calidad del sistema, fijando tiempos objetivo de respuesta inferiores a 2 segundos para operaciones CRUD y menores a 5 segundos para la carga del tablero con más de 100 tareas, además de los criterios de seguridad (JWT/OAuth 2.0 y comunicación bajo HTTPS/TLS), accesibilidad conforme a las pautas WCAG 2.1 nivel AA y trazabilidad mediante el registro de historial y auditoría de las operaciones realizadas en la plataforma.

# Etapa 2 Persistencia de Datos y Backend (Servidor)

Se implementó la capa de servicios encargada de persistir y exponer la lógica de Task Assistant. Para ello, se configuró una base de datos relacional en PostgreSQL 14+, definiendo las tablas, migraciones, restricciones (constraints) e índices necesarios para optimizar las búsquedas y el manejo de fechas. Sobre esta capa de datos se desarrolló una API REST (basada en Node.js con Express / NestJS), incorporando autenticación mediante JWT y soporte para OAuth 2.0 en caso de integraciones externas, así como control de acceso por roles, mecanismos de validación y registro de auditoría. Desde el servidor se dio cobertura a los casos de uso clave: operaciones CRUD y recuperación de tareas, asignación y distribución entre estudiantes y docentes, gestión de comentarios y menciones, administración de metas, calificaciones, exportaciones y sincronización con calendarios como Google y Outlook. Finalmente, se trabajó en los aspectos de calidad y despliegue, implementando pruebas unitarias y de integración, documentando la API con OpenAPI/Swagger, configurando un flujo de CI/CD y realizando el despliegue en la nube, con copias de seguridad periódicas y políticas de retención de la información.

# Etapa 3 — Consumo de Datos y Frontend (Cliente)

Se construyó el cliente PWA que consume la API y materializa la experiencia de usuario de Task Assistant. Para ello, se desarrolló una interfaz responsive, adaptable a desktop, tablet y dispositivos móviles, cumpliendo criterios de accesibilidad alineados con WCAG 2.1 AA, con formularios que incorporan validación en tiempo real, funcionalidades de búsqueda y filtrado, así como una vista de calendario para la organización temporal de las tareas. Se implementaron los flujos principales de uso: creación, edición y asignación de tareas; gestión de comentarios y @menciones; seguimiento del progreso y registro de tiempo; definición de metas semanales y mensuales; calificación docente; notificaciones; y exportación de información en formatos PDF y Excel. Adicionalmente, se integró la sincronización con calendarios externos, gestionada mediante autorización explícita del usuario, junto con un manejo seguro de sesiones. Finalmente, se abordaron los aspectos de calidad del cliente, realizando pruebas funcionales y de usabilidad, estableciendo y midiendo métricas de desempeño (manteniendo tiempos inferiores a 2 segundos en las operaciones más frecuentes) y configurando la monitorización de errores del lado del cliente para garantizar una experiencia estable y confiable.

# Alcance del proyecto Task Assistant

Task Assistant es una aplicación web para gestión de tareas educativas que centraliza la creación, asignación y seguimiento de actividades en contextos colaborativos. Su alcance actual comprende la planificación y organización de tareas por proyectos, la colaboración contextual mediante comentarios y @menciones, la gestión de recursos educativos (archivos y enlaces), el seguimiento de progreso y tiempo, la planificación temporal con metas semanales y mensuales, la evaluación y calificación, las notificaciones y la exportación de reportes. Incluye sincronización con calendarios externos y control de acceso por roles (Administrador, Docente Principal, Docente Invitado, Estudiante), con auditoría e historial de cambios.

**Casos de uso núcleo (25):**

1. CRUD de tareas
2. Recuperar tareas eliminadas
3. Ajustar fechas de entrega
4. Comentarios colaborativos
5. Menciones (@usuario)
6. Monitoreo de progreso
7. Distribución de tareas
8. Asignación de recursos
9. Asignación de tarea
10. Sincronizar con calendarios
11. Etiquetar por color
12. Etiquetar por palabras clave
13. Agrupar por proyecto
14. Duplicar tareas
15. Guardar cambios de tarea
16. Crear metas semanales
17. Crear metas mensuales
18. Calificar tareas completadas
19. Notificar cambios
20. Registrar tiempo por tarea
21. Exportar tareas
22. Exportar en PDF
23. Exportar en Excel
24. Importar tareas desde archivos externos
25. Anclar tareas

**Límites del alcance (exclusiones):**

* Integración con LMS externos (Moodle, Canvas, Blackboard)
* Videoconferencias o mensajería privada independiente
* Gestión de pagos o facturación
* Funcionalidades basadas en IA/ML (p. ej., predicciones o asistentes de voz) ● Aplicaciones móviles nativas (se ofrece PWA responsive)

**Funcionalidades futuras:**

* Personalización de vistas del tablero (lista, kanban, timeline/Gantt)
* Filtrado avanzado por múltiples criterios (estado, etiquetas, fechas, asignados)
* Programación de tareas recurrentes (diarias/semanales/mensuales)
* Reasignación rápida desde tablero
* Gráficas de rendimiento y cumplimiento de objetivos
* Integración con almacenamiento en la nube (Drive, Dropbox)
* Predicción de fechas óptimas y prioridades mediante ML
* Asistente por voz/chat con IA para creación/edición/consulta
* Recomendaciones de pausas y detección de sobrecarga
* Detección de conflictos de agenda entre tareas y eventos
* Gamificación (logros, medallas, niveles)
* Alertas preventivas por riesgo de retraso
* Reglas automatizadas personalizadas (p. ej., avisos externos)
* Dashboards de bienestar digital (tiempo frente a pantalla, dispersión)

# Definiciones y Acrónimos

**API:** Interfaz de Programación de Aplicaciones (Application Programming Interface).

**REST:** Transferencia de Estado Representacional (Representational State Transfer).

**API RESTful:** API que implementa los principios de REST.

**HTTP/HTTPS:** Protocolo de Transferencia de Hipertexto / HTTP seguro.

**TLS / HSTS:** Seguridad de la Capa de Transporte (Transport Layer Security) / Strict Transport Security.

**JSON:** Notación de Objetos de JavaScript (JavaScript Object Notation).

**JWT:** JSON Web Token para autenticación y autorización.

OAuth 2.0 / SSO: Marco de autorización / Inicio de sesión único (Single Sign-On).

**RBAC:** Control de Acceso Basado en Roles (Role-Based Access Control). **CORS:** Intercambio de Recursos de Origen Cruzado (Cross-Origin Resource Sharing).

**WebSocket / SSE:** Canal bidireccional en tiempo real / Server-Sent Events para actualizaciones.

**PWA:** Aplicación Web Progresiva (Progressive Web App).

**SPA**: Aplicación de Página Única (Single Page Application).

**UI / UX:** Interfaz de Usuario / Experiencia de Usuario.

**WCAG 2.1:** Pautas de Accesibilidad para el Contenido Web (nivel AA).

**CRUD:** Crear, Leer, Actualizar y Borrar (Create, Read, Update, Delete).

**DBMS / RDBMS**: Sistema (Relacional) de Gestión de Bases de Datos.

**SQL:** Lenguaje de Consulta Estructurada (Structured Query Language).

**ORM:** Mapeo Objeto-Relacional (Object-Relational Mapping).

**PK / FK:** Clave primaria / clave foránea (Primary Key / Foreign Key).

**3FN:** Tercera Forma Normal (Third Normal Form).

**UUID:** Identificador Único Universal (Universally Unique Identifier).

**OpenAPI / Swagger:** Especificación y herramientas para documentar APIs. **CI/CD:** Integración Continua / Entrega Continua (Continuous Integration / Continuous Delivery).

**HTML / CSS / JS:** Lenguaje de marcado / hojas de estilo / JavaScript.

**DOM:** Modelo de Objeto del Documento (Document Object Model).

**IDE / CLI**: Entorno de Desarrollo Integrado / Interfaz de Línea de Comandos.

**2. Descripción General**

# Objetivos del Sistema

El objetivo de Task Assistant es proporcionar una plataforma integral de gestión de tareas educativas que permita a docentes y estudiantes organizar, ejecutar y monitorear actividades académicas de manera eficiente y colaborativa. Los objetivos específicos son:

# Objetivos funcionales

* Centralizar la gestión de tareas académicas en una plataforma única y accesible.
* Facilitar la asignación y distribución equitativa de actividades entre estudiantes.
* Promover la colaboración mediante comentarios, menciones y trabajo en equipo.
* Proporcionar visibilidad del progreso individual y grupal en tiempo real.
* Permitir la evaluación y calificación de tareas con retroalimentación estructurada.
* Ofrecer herramientas de planificación temporal mediante metas y calendarios.

# Objetivos operacionales

* Reducir el tiempo invertido en la gestión administrativa de tareas.
* Mejorar la comunicación entre docentes y estudiantes sobre actividades asignadas.
* Aumentar la transparencia en el seguimiento de avances académicos.
* Facilitar la documentación y exportación de evidencias de trabajo.
* Optimizar la organización de proyectos educativos de mediano y largo plazo.

# Objetivos de experiencia de usuario

* Proporcionar una interfaz intuitiva que requiere capacitación mínima.
* Garantizar accesibilidad desde múltiples dispositivos (escritorio, móvil, tablet). ● Ofrecer notificaciones oportunas sin generar sobrecarga informativa.
* Personalizar la experiencia según el rol del usuario (Administrador, Docente, Estudiante).
* Mantener tiempos de respuesta inferiores a 2 segundos en operaciones comunes.

# Funcionalidad General

* **Gestión integral de tareas:** creación, edición, asignación, duplicación y eliminación con recuperación desde papelera.
* **Sistema de asignación:** distribución manual o automática de tareas a estudiantes y grupos, con notificación inmediata.
* **Colaboración contextual**: comentarios por tarea y @menciones para conversaciones centradas en la actividad.
* **Recursos educativos:** adjuntar archivos (PDF, imágenes, videos, documentos) y enlaces web con control básico de versiones.
* **Seguimiento y monitoreo:** estados (nueva, en progreso, completada), porcentaje de avance y registro de tiempo por usuario.
* **Organización flexible:** agrupación por proyectos, etiquetas por color y palabras clave, y anclado de tareas prioritarias.
* **Planificación temporal:** metas semanales y mensuales, ajuste de fechas de entrega y sincronización con calendarios externos (Google Calendar, Outlook).
* **Evaluación:** calificación numérica o cualitativa con notas de retroalimentación y trazabilidad.
* **Notificaciones:** alertas por nuevas asignaciones, cambios relevantes, menciones y proximidad de fechas límite.
* **Exportaciones:** generación de reportes en PDF y Excel con filtros previos de selección.
* **Control de acceso por roles:** permisos diferenciados para Administrador, Docente Principal, Docente Invitado y Estudiante.
* **Historial de cambios:** registro automático de modificaciones para auditoría y seguimiento.
* **Interfaz adaptable:** diseño responsive optimizado para escritorio, tablet y móvil.
* **Búsqueda y filtros avanzados:** localización de tareas por estado, etiqueta, fecha, proyecto y asignado.
* **Panel administrativo:** vista centralizada de usuarios, proyectos, métricas de productividad y estadísticas de uso.

# Usuarios del Sistema

Los siguientes usuarios pueden interactuar con la pizarra dependiendo de las funcionalidades:

✓ = Permiso completo · ✓\* = Solo sobre sus propias tareas/datos · — = Sin permiso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Funcionalidad** | **Administrador** | **Docente**  **Principal** | **Docente**  **Invitado** | **Estudiante** |
| CRUD Tarea | ✓ | ✓ | — | — |
| Recuperar Tarea  (papelera) | ✓ | ✓ | — | — |
| Ajustar Fechas de  Entrega | ✓ | ✓ | — | — |
| Comentarios  Colaborativos | ✓ | ✓ | ✓ | ✓\* |
| Menciones @usuario | ✓ | ✓ | ✓ | ✓ |
| Monitorear Progreso | ✓ | ✓ | ✓ | ✓\* |
| Distribuir Tareas | ✓ | ✓ | — | — |
| Asignar Recursos | ✓ | ✓ | — | — |
| Asignar Tarea | ✓ | ✓ | — | — |
| Sincronizar con  Calendarios | ✓ | ✓ | ✓ | ✓\* |
| Etiquetar por Color | ✓ | ✓ | ✓\* | ✓\* |
| Etiquetar por  Palabras Clave | ✓ | ✓ | ✓\* | ✓\* |
| Agrupar por Proyecto | ✓ | ✓ | ✓ | ✓\* |
| Duplicar Tareas | ✓ | ✓ | ✓\* | ✓\* |
| Guardar Cambios de  Tarea | ✓ | ✓ | — | — |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Crear Metas  Semanales | ✓ | ✓ | ✓\* | ✓\* |
| Crear Metas  Mensuales | ✓ | ✓ | — | — |
| Calificar Tareas  Completadas | ✓ | ✓ | — | — |
| Notificar Cambios | ✓ | ✓ | ✓ | ✓ |
| Registrar Tiempo por  Tarea | ✓ | ✓ | ✓\* | ✓\* |
| Exportar Tarea | ✓ | ✓ | ✓\* | ✓\* |
| Exportar en PDF | ✓ | ✓ | ✓\* | ✓\* |
| Exportar en Excel | ✓ | ✓ | ✓\* | ✓\* |
| Anclar Tareas | ✓ | ✓ | ✓\* | ✓\* |

# Restricciones

**Acceso por invitación y rol:** Solo usuarios registrados e invitados por Administrador o Docente Principal pueden acceder a los proyectos y funcionalidades.

**Alcance por proyecto:** La visibilidad y las acciones de cada usuario se limitan a los proyectos en los que participa (no hay acceso global).

**Docente Invitado:** Puede colaborar (comentarios, menciones, seguimiento, algunas exportaciones acotadas), pero no crear/eliminar/asignar tareas, no ajustar fechas, no recuperar desde papelera, no calificar.

**Estudiante:** Solo actúa sobre sus tareas y metas personales; no asigna tareas, no modifica fechas de otros, no elimina/recupera tareas, no califica, no gestiona recursos de terceros.

**Recuperación de eliminadas:** Disponible únicamente para Administrador y Docente Principal, con retención en papelera de 30 días.

**Sincronización con calendarios:** Requiere autorización individual; el alcance se restringe al propietario; los tokens se cifran y pueden revocarse en cualquier momento.

**Exportaciones:** Admin/Docente Principal pueden exportar proyectos completos; Docente Invitado/Estudiante solo exportan datos de su propio alcance.

**Archivos adjuntos:** Tamaño máximo 50 MB por archivo y tipos permitidos definidos por política; se aplican controles de seguridad.

**Seguridad de sesión:** Acceso únicamente por HTTPS; autenticación JWT vigente; cierre automático por inactividad a los 30 min.

**Notificaciones y límites de uso:** Envío sujeto a consentimiento del usuario y a políticas de frecuencia para evitar sobrecarga.

## 3. Requisitos funcionales

### Gestión de tareas

**CRUD de tareas:** El sistema debe permitir crear, leer, actualizar y eliminar tareas (solo Administrador y Docente Principal).

**Recuperación:** El sistema debe permitir restaurar tareas eliminadas desde una papelera con retención de 30 días (Administrador y Docente Principal).

**Ajuste de fechas:** El sistema debe permitir definir y modificar la fecha/hora de entrega de una tarea (Administrador y Docente Principal).

**Duplicación:** El sistema debe permitir duplicar una tarea preservando campos seleccionados y generando un nuevo identificador.

**Guardar cambios:** El sistema debe persistir ediciones de una tarea y registrar fecha, autor y detalle del cambio.

### Asignación y distribución

**Asignar tarea:** El sistema debe permitir asignar una tarea a uno o varios estudiantes o grupos (Administrador y Docente Principal).

**Distribución:** El sistema debe permitir distribuir tareas de forma manual o automática según reglas de carga configurables (Administrador y Docente Principal).

### Colaboración

**Comentarios:** El sistema debe permitir publicar, editar y eliminar comentarios propios en la tarea (todos los roles; el Estudiante solo en tareas donde participa).

**Menciones:** El sistema debe permitir mencionar usuarios con @usuario y notificar al mencionado si tiene acceso al proyecto.

### Recursos educativos

**Adjuntos y enlaces:** El sistema debe permitir adjuntar archivos (≤ 50 MB por archivo; tipos permitidos por política) y agregar enlaces a una tarea, con registro de versión del recurso.

### Seguimiento y progreso

**Estados y porcentajes:** El sistema debe permitir cambiar el estado de la tarea (nueva, en progreso, completada, archivada) y actualizar el porcentaje de avance (0–100).

**Registro de tiempo:** El sistema debe permitir registrar tiempo dedicado por usuario a una tarea (todos los roles; el Estudiante solo en sus tareas).

### Organización

**Proyectos:** El sistema debe permitir agrupar tareas por proyecto y cambiar su pertenencia manteniendo la trazabilidad.

**Etiquetas por color:** El sistema debe permitir etiquetar tareas con colores predefinidos.

**Etiquetas por palabras clave:** El sistema debe permitir añadir etiquetas libres (búsqueda por texto).

**Anclado:** El sistema debe permitir anclar tareas para priorizarlas en vistas personales.

#### 3.7 Planificación

**Metas semanales:** El sistema debe permitir crear y gestionar metas semanales, vinculando tareas y mostrando cumplimiento.

**Metas mensuales:** El sistema debe permitir crear y gestionar metas mensuales con submetas.

**Sincronización con calendarios:** El sistema debe permitir sincronizar tareas y vencimientos con Google Calendar y Outlook por usuario autorizado.

#### 3.8 Evaluación

**Calificación:** El sistema debe permitir calificar tareas (escala 0–100 o cualitativa) y emitir retroalimentación textual (Docente Principal).

#### 3.9 Notificaciones y reportes

**Notificaciones**: El sistema debe notificar eventos relevantes (asignaciones, cambios, menciones, recordatorios de vencimiento) según preferencias del usuario.

**Exportar tareas:** El sistema debe permitir exportar conjuntos de tareas seleccionadas con filtros.

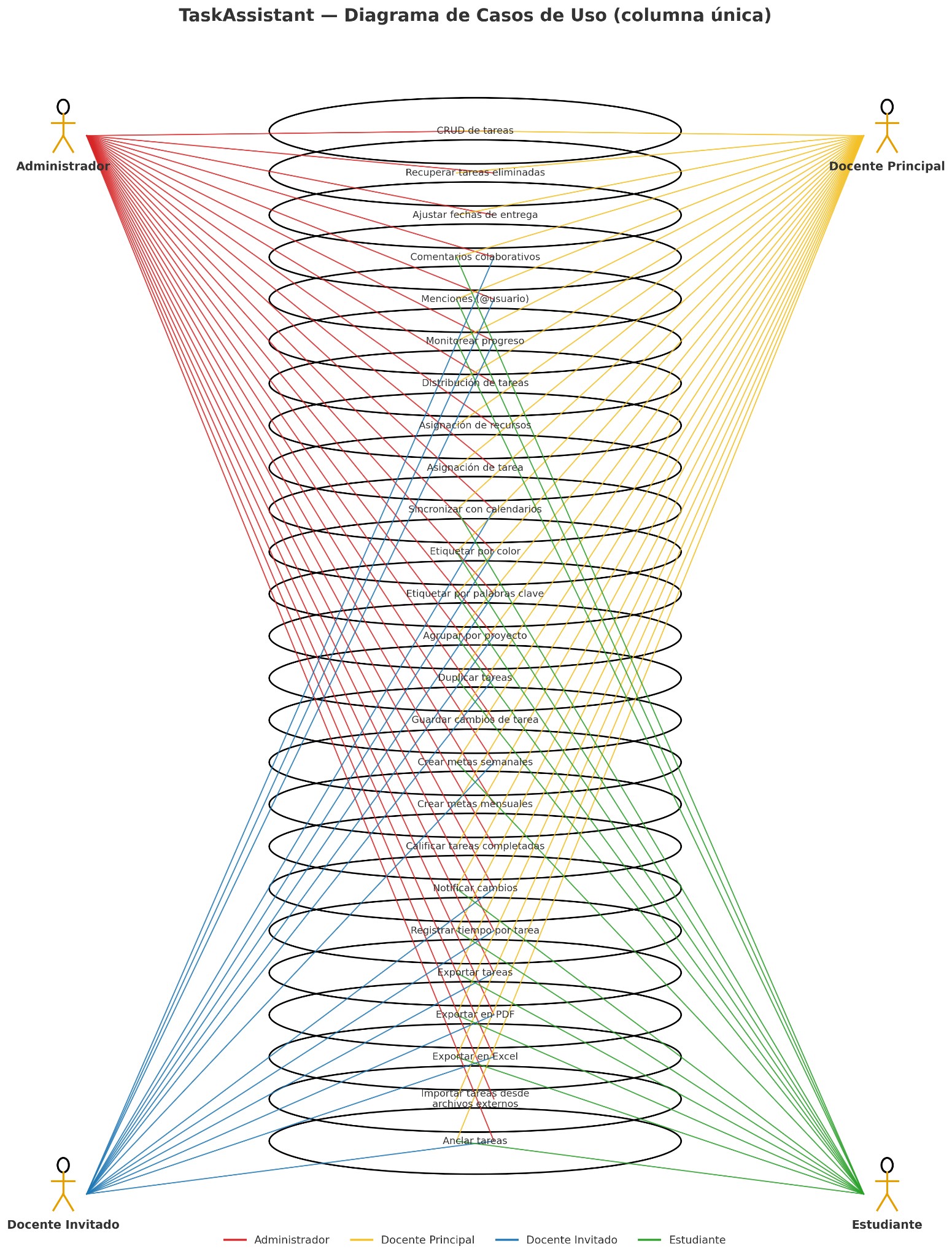
**Exportar PDF:** El sistema debe permitir generar PDF con formato de reporte.

**Exportar Excel:** El sistema debe permitir generar archivos .xlsx con estructura tabular.

**Importar tareas:** El sistema debe permitir importar tareas desde archivos externos (CSV/Excel) con validación y previsualización.

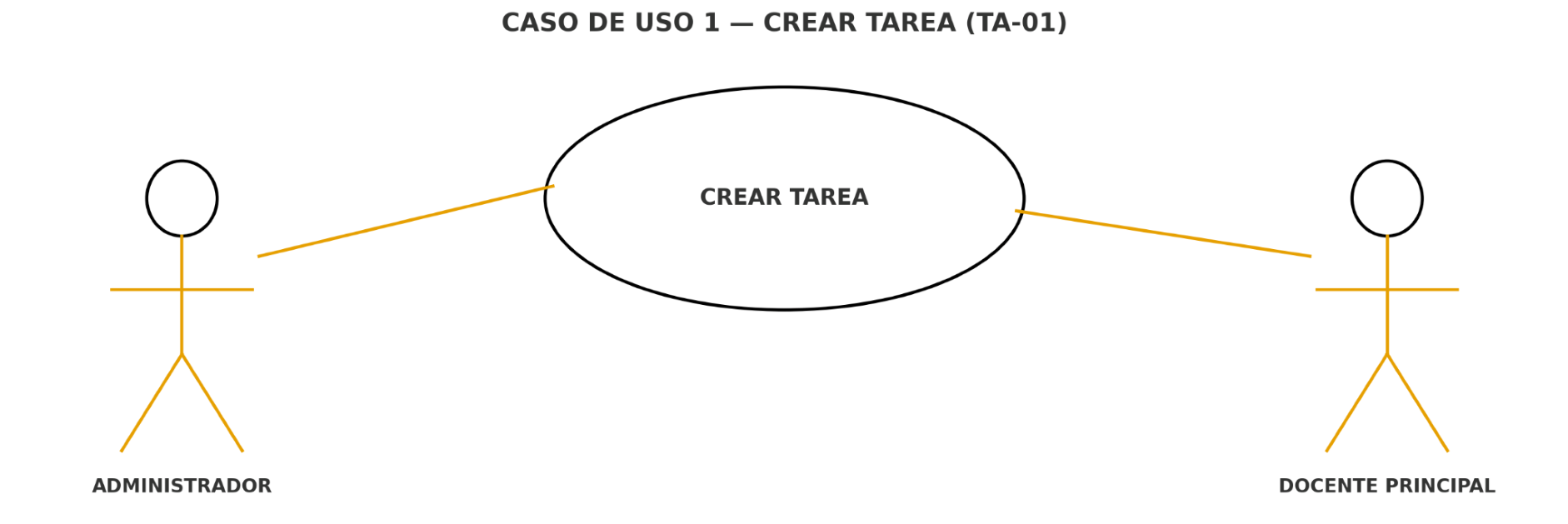
# Casos de uso

Diagrama de caso de uso



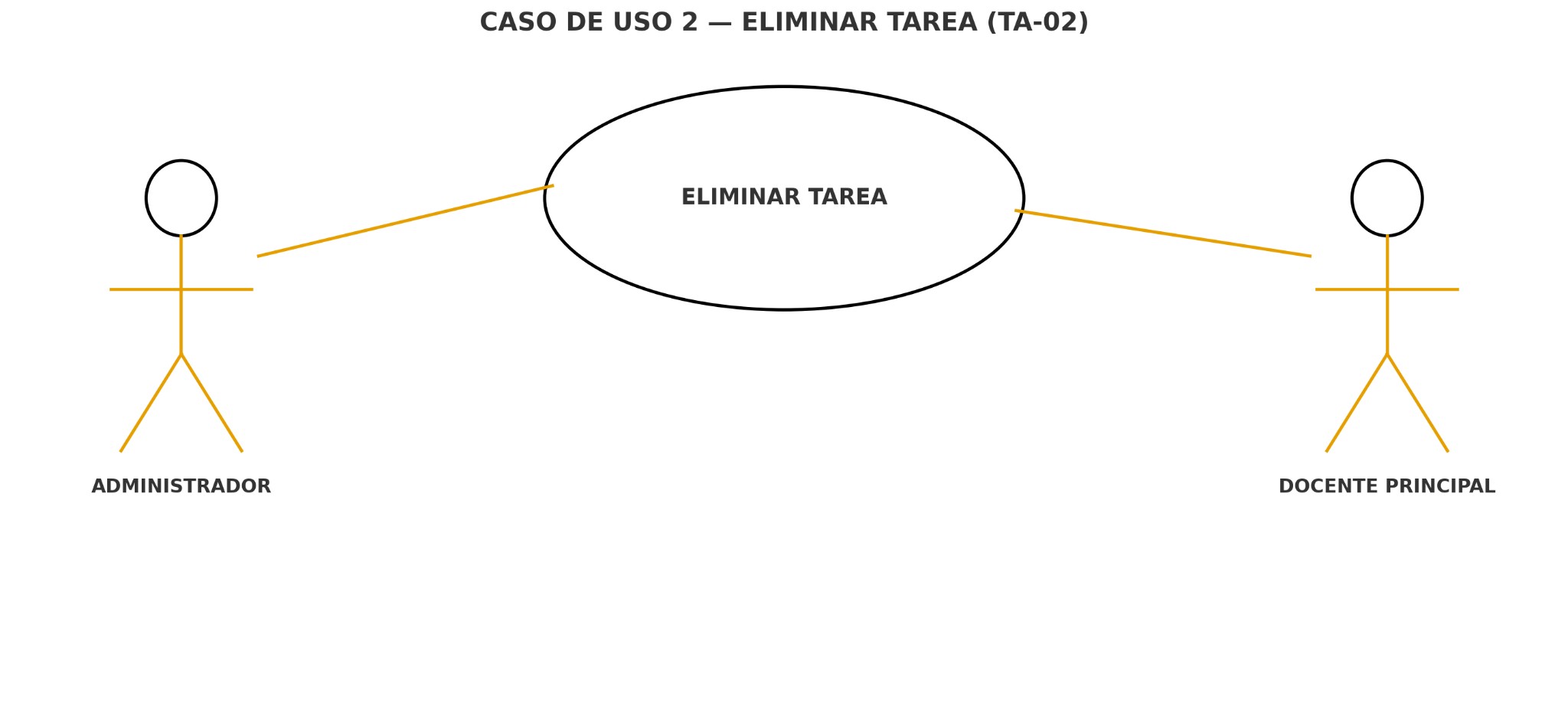
## Crear Tarea

|  |  |  |
| --- | --- | --- |
|  | TA-1 | |
| **Nombre** | Crear tarea | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Registrar una nueva tarea en un proyecto con metadatos básicos. | |
| Urgencia | 5 | |
| Esfuerzo | 4 | |
| **Pre-condiciones** | Usuario autenticado con rol Admin o DP.  Proyecto existente y activo.  Conectividad con el backend. | |
| **Flujo Normal** | Usuario | Sistema |
| Abre “Nueva tarea”. |  |
|  | Muestra formulario con campos obligatorios. |
| Completa datos (título, proyecto, fechas, prioridad). |  |
|  | Valida entradas y habilita “Crear”. |
| Confirma la creación. |  |
|  | Persiste la tarea, registra auditoría y muestra confirmación. |
| **Post-condiciones** | Tarea creada y visible en las vistas del proyecto. Registro en historial de cambios. | |
| **Excepciones** | E1 Datos inválidos: El sistema marca campos y no guarda.  E2 Error de red: Reintento y mensaje de indisponibilidad temporal. | |



## Eliminar Tarea

|  |  |  |
| --- | --- | --- |
|  | TA-2 | |
| **Nombre** | Eliminar Tarea | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Enviar una tarea a papelera para ocultarla de las vistas activas. | |
| Urgencia | 4 | |
| Esfuerzo | 2 | |
| **Pre-condiciones** | Usuario con permisos de edición sobre la tarea.  Tarea en estado distinto de “archivada” (o política que lo permita). | |
| **Flujo Normal** | Usuario | Sistema |
| Selecciona “Eliminar”. |  |
|  | Solicita confirmación. |
| Confirma eliminación. |  |
|  | Mueve a la papelera, registra auditoría y muestra confirmación. |
| **Post-condiciones** | Tarea en la papelera con marca de tiempo. No aparece en vistas activas. | |
| **Excepciones** | E1 Sin permisos: Mensaje “No autorizado”.  E2 Conflicto de estado: Mensaje de restricción y no se elimina. | |



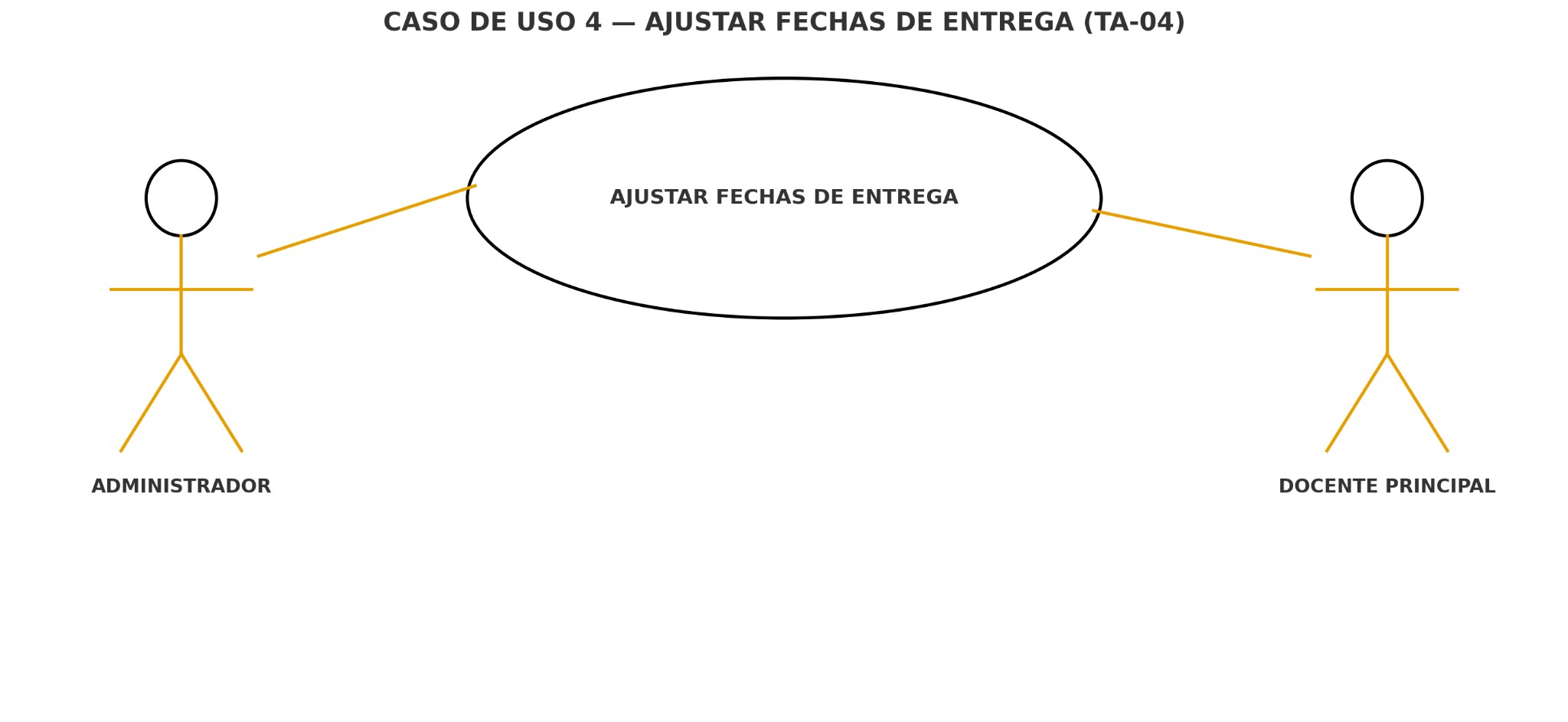
## Recuperar Tarea Eliminada

|  |  |  |
| --- | --- | --- |
|  | TA-3 | |
| **Nombre** | Recuperar Tarea Eliminada | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Restaurar una tarea desde la papelera dentro del periodo de retención. | |
| Urgencia | 4 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Tarea en papelera y dentro de retención (p. ej., 30 días).  Usuario con permisos. | |
| **Flujo Normal** | Usuario | Sistema |
| Abre “Papelera” y selecciona la tarea. |  |
|  | Muestra la opción “Restaurar”. |
| Confirma restauración.. |  |
|  | Restituye la tarea y confirma. |
| **Post-condiciones** | Tarea visible nuevamente en el proyecto. Auditoría de restauración registrada. | |
| **Excepciones** | E1 Retención vencida: Mensaje y opción no disponible. E2 Sin permisos: Mensaje “No autorizado”. | |



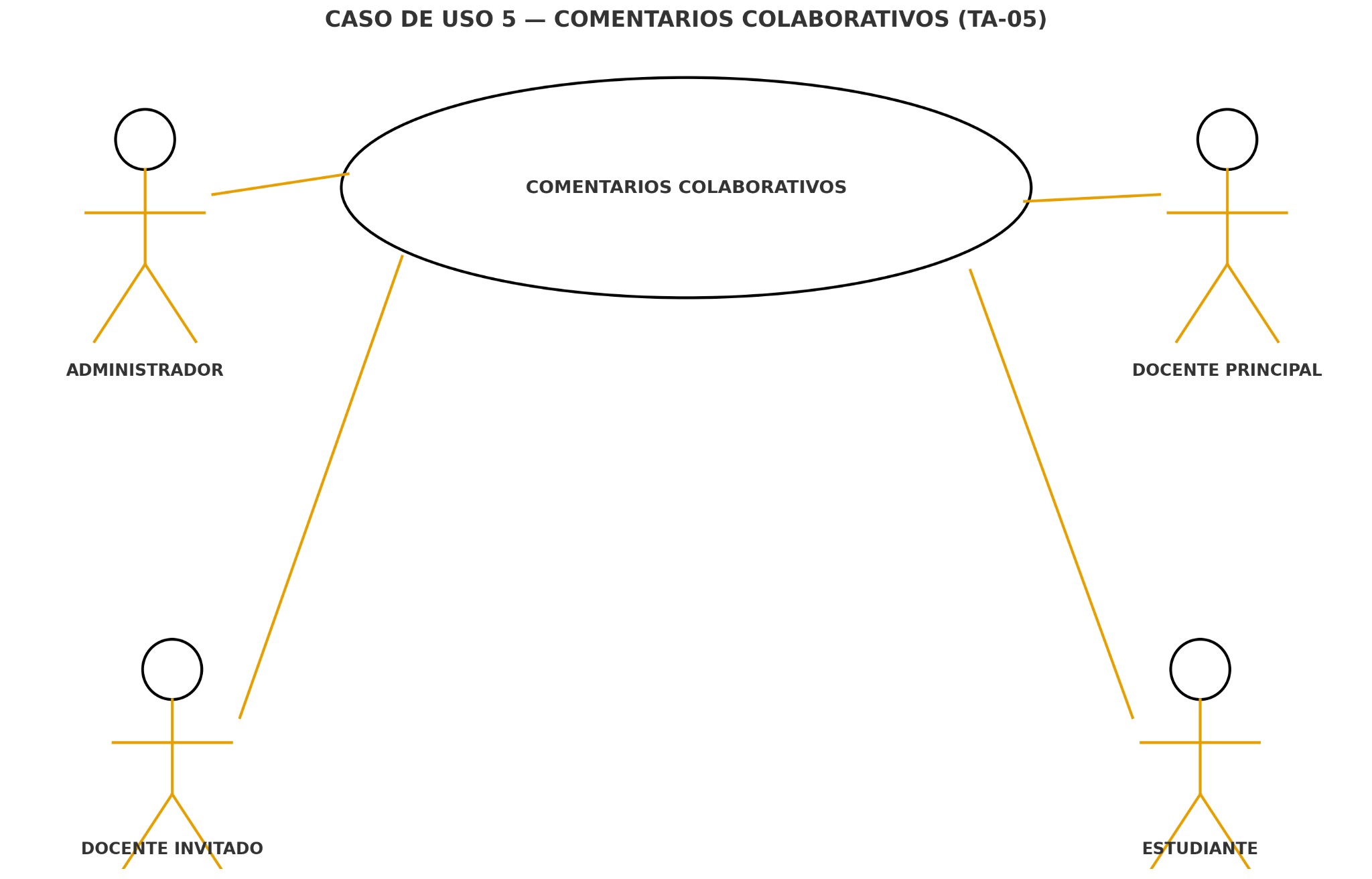
## Ajustar Fechas de Entrega

|  |  |  |
| --- | --- | --- |
|  | TA-4 | |
| **Nombre** | Ajustar Fechas de Entrega | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Modificar fecha/hora de entrega respetando validaciones. | |
| Urgencia | 4 | |
| Esfuerzo | 2 | |
| **Pre-condiciones** | Tarea existente.  Permisos de edición. | |
| **Flujo Normal** | Usuario | Sistema |
| Edita fecha/hora de entrega. |  |
|  | Válida coherencia (no menor a inicio, etc.). |
| Guarda cambios. |  |
|  | Persiste y notifica a los asignados. |
| **Post-condiciones** | Fecha/hora actualizadas.  Notificaciones emitidas según preferencias. | |
| **Excepciones** | E1 Fecha incoherente: Se bloquea guardado y se indica el error. | |



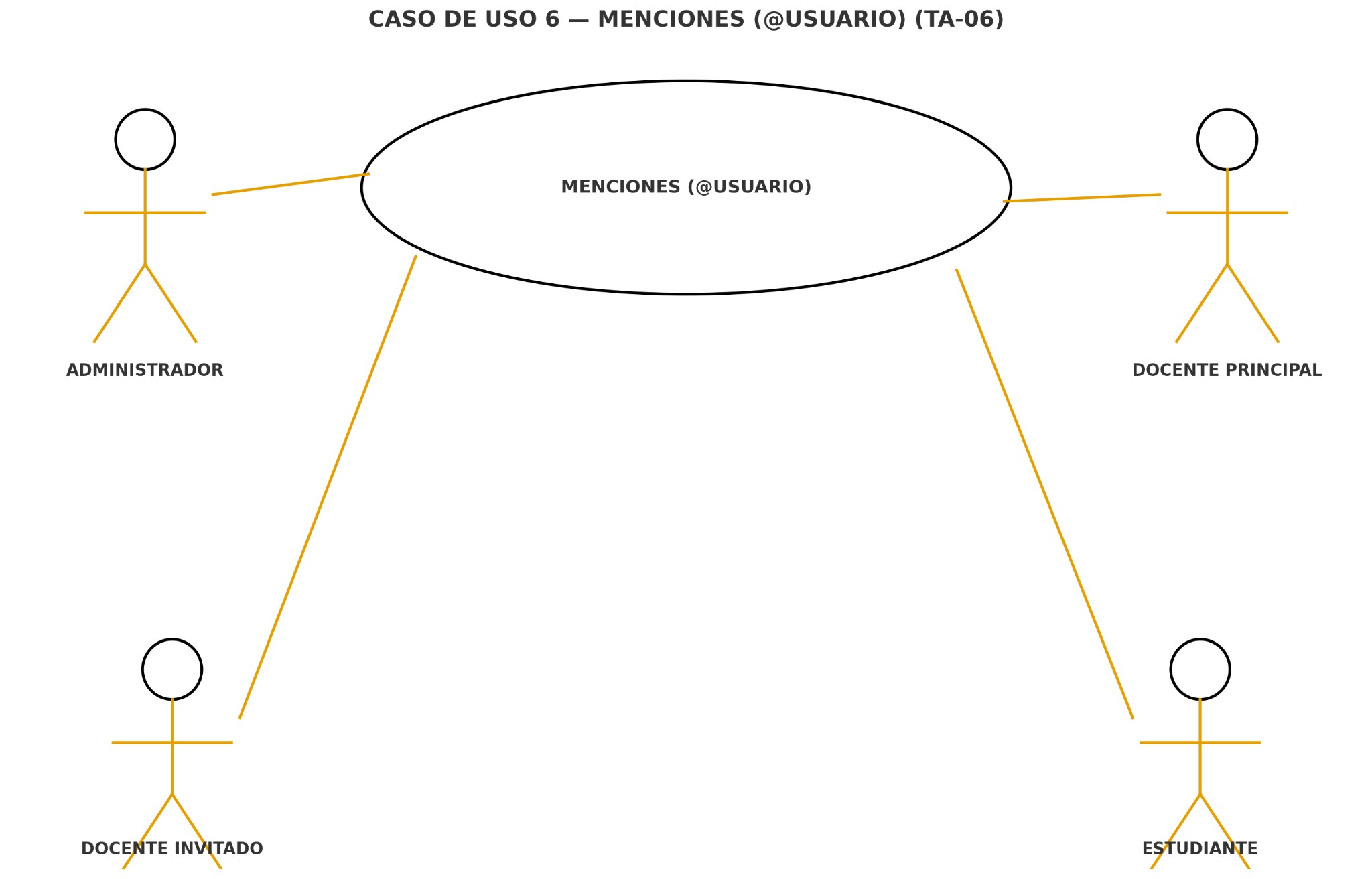
## Comentarios Colaborativos

|  |  |  |
| --- | --- | --- |
|  | TA-5 | |
| **Nombre** | Comentarios Colaborativos | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Permitir discusiones contextuales por tarea. | |
| Urgencia | 4 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Usuario con acceso al proyecto/tarea.  Tarea existente. | |
| **Flujo Normal** | Usuario | Sistema |
| Abre el panel de comentarios. |  |
|  | Carga de hilo existente. |
| Publica/edita/elimina su comentario. |  |
|  | Guarda, muestra en hilo y notifica a suscriptores |
| **Post-condiciones** | Comentario visible con autor y marca temporal. Historial conservado. | |
| **Excepciones** | E1 Contenido vacío/no permitido: Validación y no publica. | |



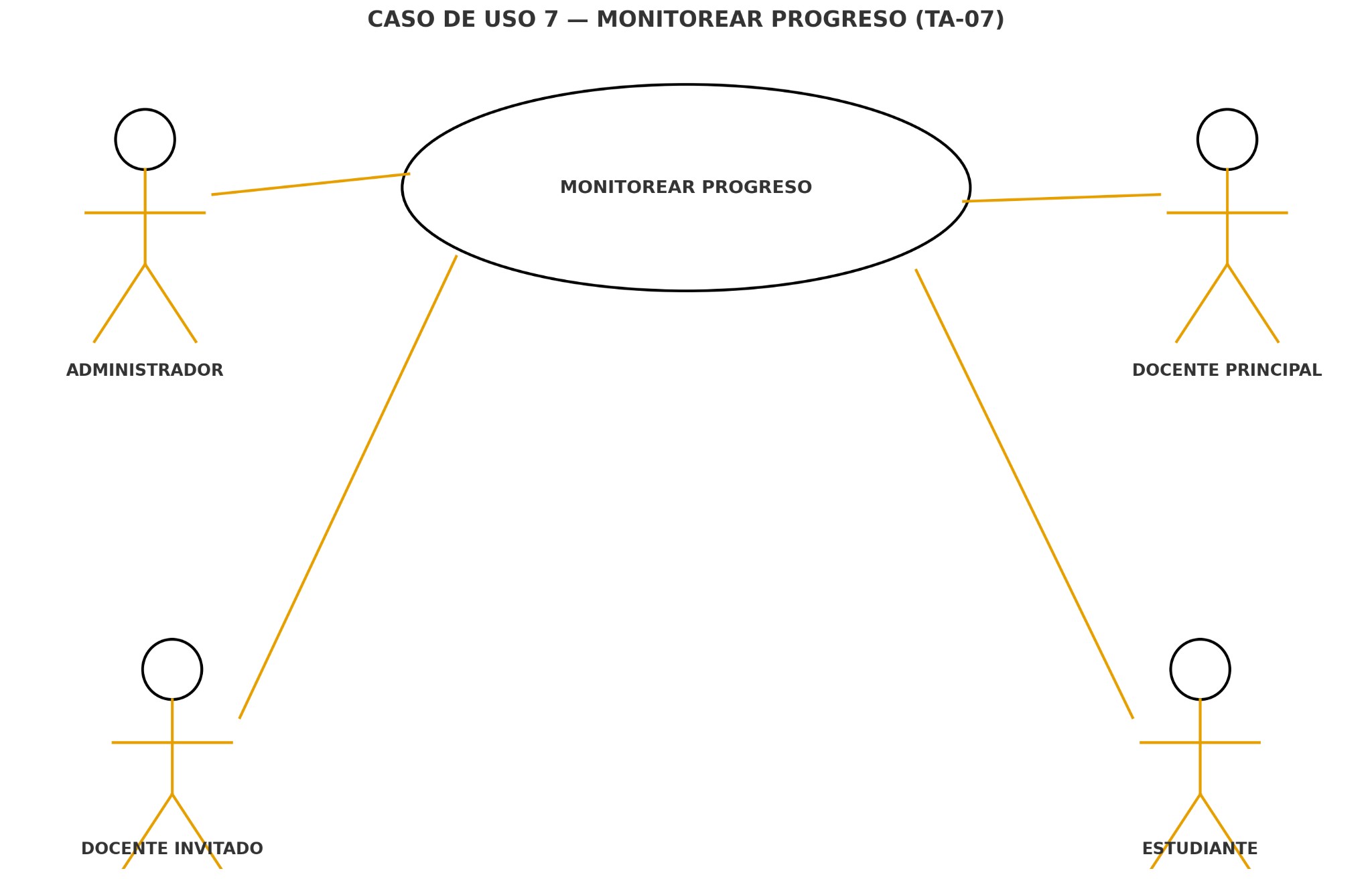
## Menciones (@usuario)

|  |  |  |
| --- | --- | --- |
|  | TA-6 | |
| **Nombre** | Menciones (@usuario) | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Notificar a un usuario referenciado en un comentario. | |
| Urgencia | 4 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Usuario mencionado con acceso al proyecto.  Comentario válido. | |
| **Flujo Normal** | Usuario | Sistema |
| Escribe “@nombre” en comentario. |  |
|  | Sugerencias de usuarios válidos. |
| Envía comentario. |  |
|  | Notifica al mencionado. |
| **Post-condiciones** | Referencia persistida.  Notificación registrada. | |
| **Excepciones** | E1 Usuario sin acceso: No disponible en sugerencias. | |



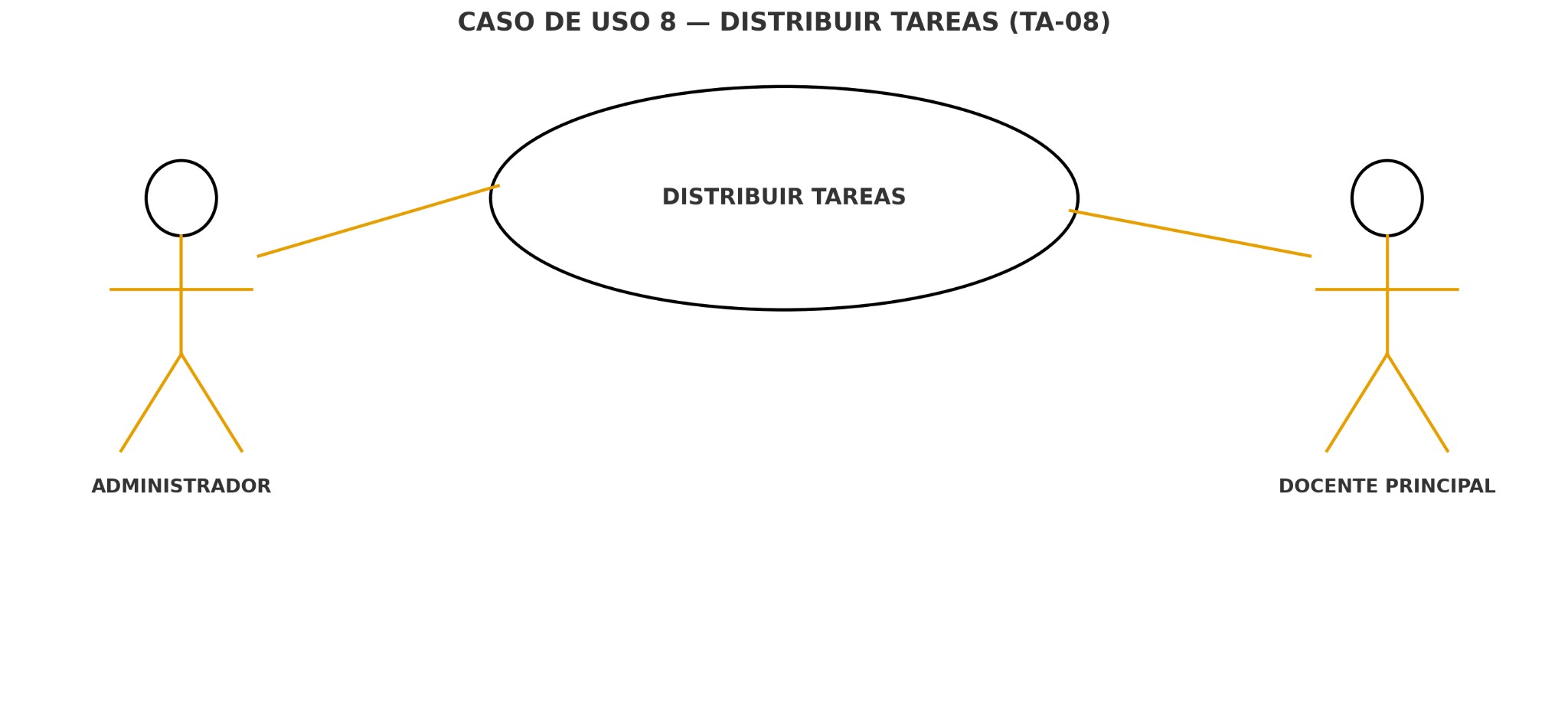
## Monitorear Progreso

|  |  |  |
| --- | --- | --- |
|  | TA-7 | |
| **Nombre** | Monitorear Progreso | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Consultar estados y % de avance de la tarea. | |
| Urgencia | 4 | |
| Esfuerzo | 2 | |
| **Pre-condiciones** | Acceso al proyecto/tarea. | |
| **Flujo Normal** | Usuario | Sistema |
| Abre la tarea. |  |
|  | Muestra estado, % avance, tiempo registrado y responsables. |
| **Post-condiciones** | Visualización actualizada. | |
| **Excepciones** | E1 Sin permisos: Acceso denegado. | |



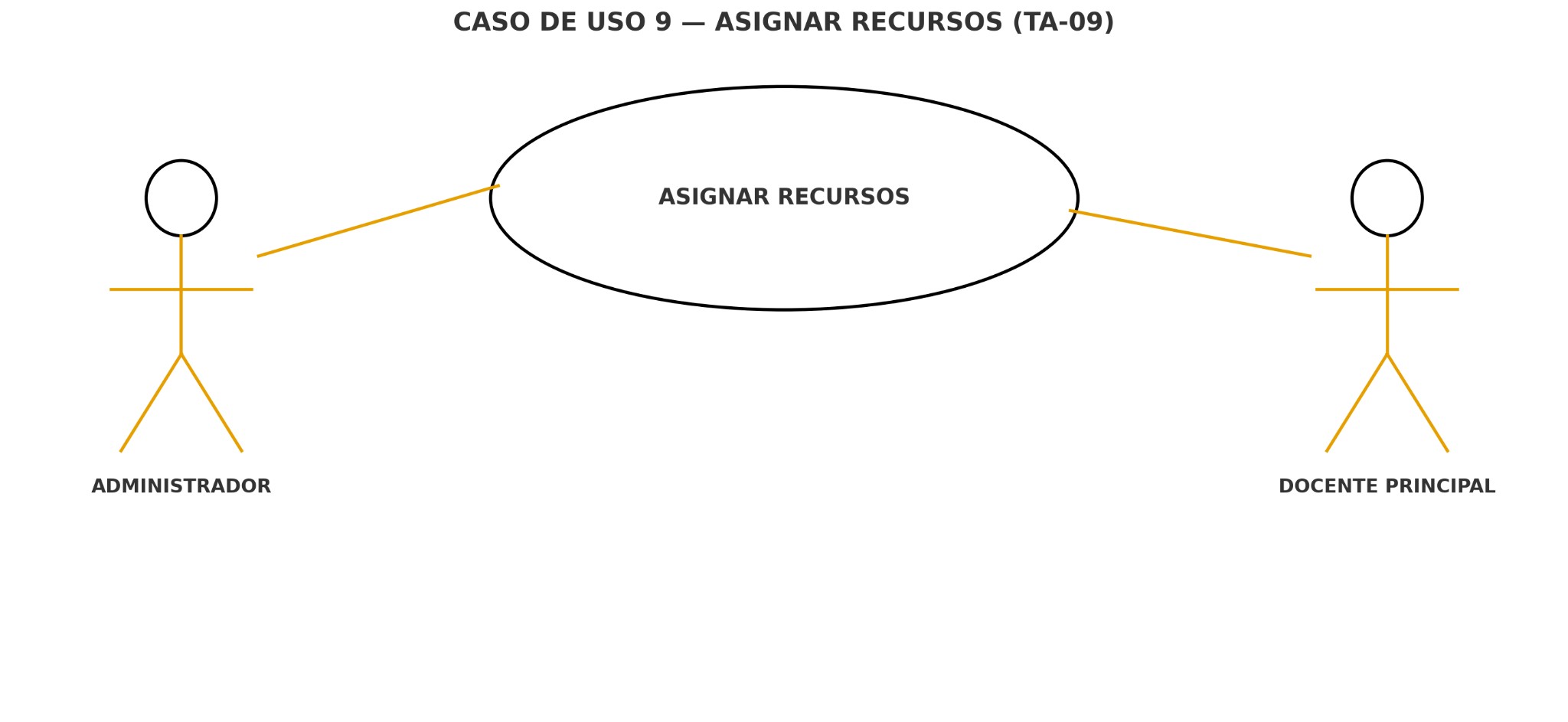
## Distribuir Tareas

|  |  |  |
| --- | --- | --- |
|  | TA-8 | |
| **Nombre** | Distribuir Tareas | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Asignar en lote tareas por reglas de carga o manualmente. | |
| Urgencia | 3 | |
| Esfuerzo | 4 | |
| **Pre-condiciones** | Conjunto de tareas seleccionadas.  Estudiantes/grupos disponibles. | |
| **Flujo Normal** | Usuario | Sistema |
| Elige modo  (manual/automático). |  |
|  | Muestra candidatos y reglas (p. ej., carga actual). |
| Confirma distribución. |  |
|  | Aplica asignaciones y notifica. |
| **Post-condiciones** | Tareas asignadas.  Auditoría de distribución. | |
| **Excepciones** | E1 Sin candidatos válidos: Mensaje y no aplica cambios. | |



## Asignar Recursos

|  |  |  |
| --- | --- | --- |
|  | TA-9 | |
| **Nombre** | Asignar Recursos | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Adjuntar archivos/enlaces a una tarea. | |
| Urgencia | 3 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Tipos y tamaño permitidos (p. ej., ≤50 MB).  Permisos de edición. | |
| **Flujo Normal** | Usuario | Sistema |
| Selecciona “Adjuntar”. |  |
|  | Valida tipo/tamaño. |
| Confirma subida o agrega URL. |  |
|  | Guarda recurso y versión, muestra en la tarea. |
| **Post-condiciones** | Recurso disponible y versionado. | |
| **Excepciones** | E1 Archivo supera límite: Bloqueo y mensaje. | |



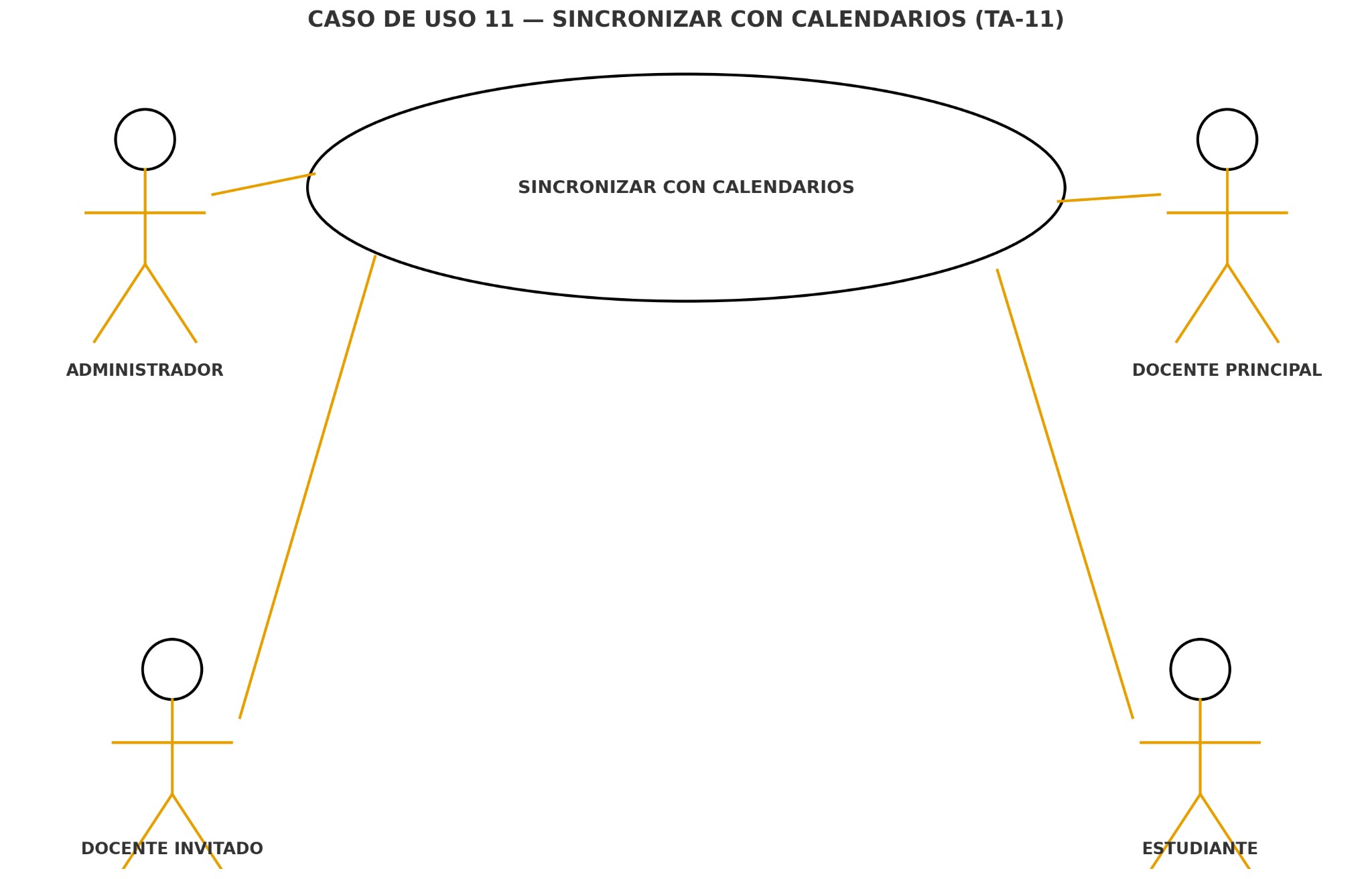
## Asignar Tarea

|  |  |  |
| --- | --- | --- |
|  | TA-10 | |
| **Nombre** | Asignar Tarea | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Designar responsables (estudiantes/grupos). | |
| Urgencia | 5 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Tarea existente.  Destinatarios con acceso al proyecto. | |
| **Flujo Normal** | Usuario | Sistema |
| Abre “Asignar”. |  |
|  | Lista estudiantes/grupos. |
| Selecciona destinatarios y confirma. |  |
|  | Persiste asignaciones y notifica. |
| **Post-condiciones** | Asignaciones registradas. | |
| **Excepciones** | E1 Usuario no elegible: Se excluye y se informa. | |



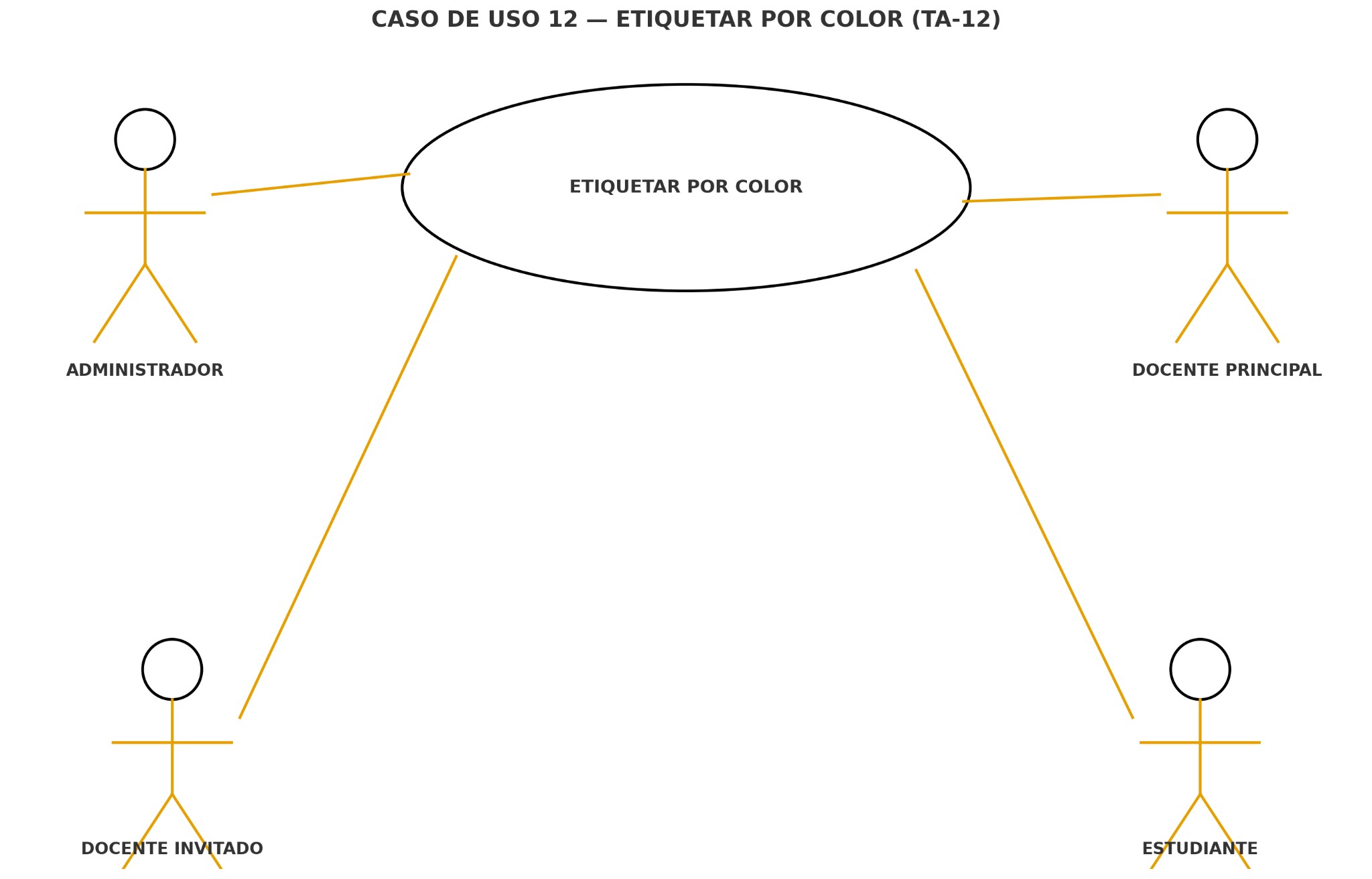
## Sincronizar con Calendarios

|  |  |  |
| --- | --- | --- |
|  | TA-11 | |
| **Nombre** | Sincronizar con Calendarios | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Sincronizar vencimientos con Google/Outlook por usuario.. | |
| Urgencia | 3 | |
| Esfuerzo | 4 | |
| **Pre-condiciones** | Autorización del usuario con el proveedor.  Conectividad con API externa. | |
| **Flujo Normal** | Usuario | Sistema |
| Autoriza proveedor (OAuth). |  |
|  | Guarda token seguro. |
| Activa sincronización. |  |
|  | Publica/actualiza eventos y confirma. |
| **Post-condiciones** | Eventos creados/actualizados en el calendario del usuario. | |
| **Excepciones** | E1 Token revocado/expirado: Solicitar reautorización. | |



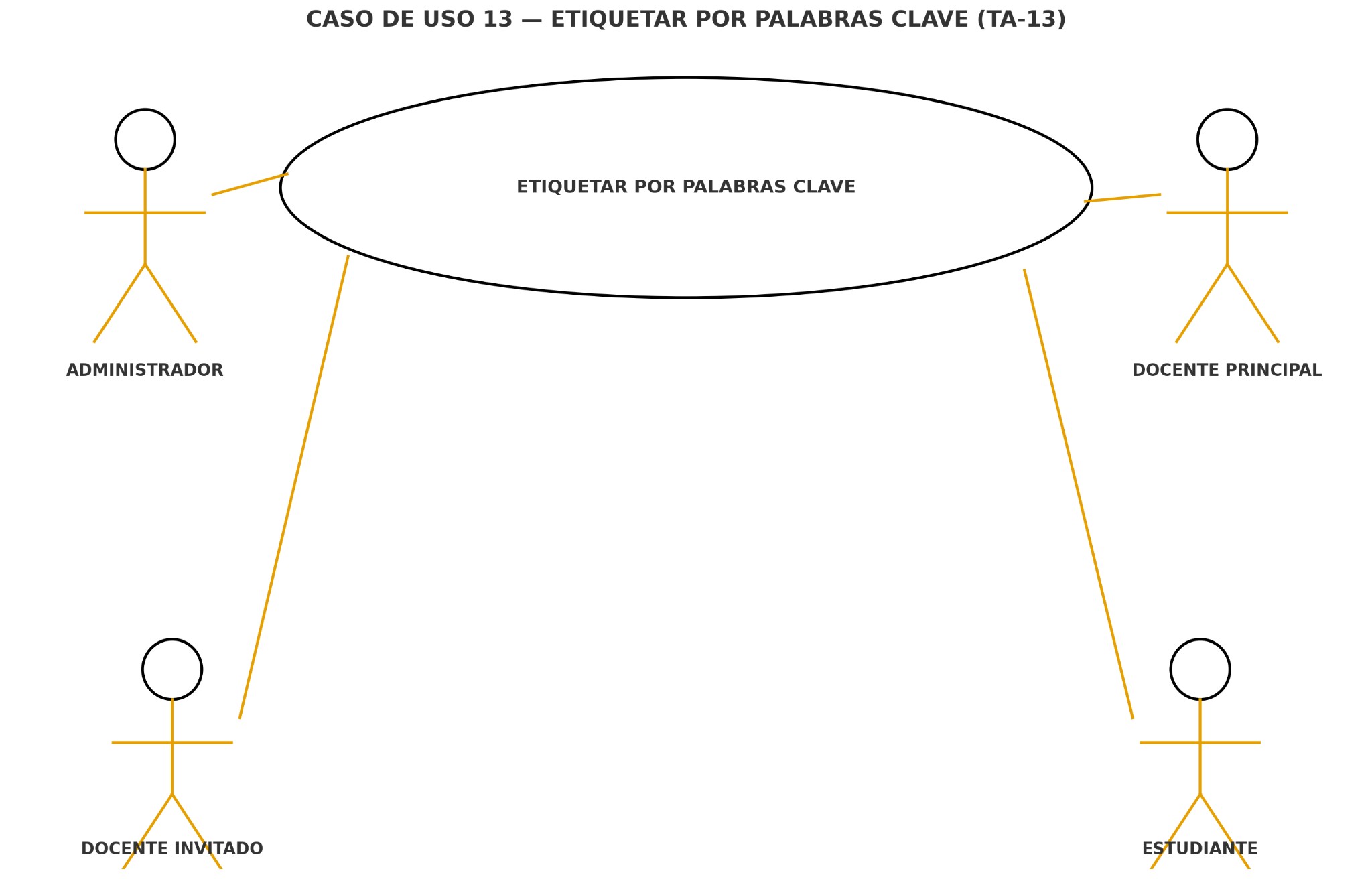
## Etiquetar por Color

|  |  |  |
| --- | --- | --- |
|  | TA-12 | |
| **Nombre** | Etiquetar por Color | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Asignar colores predefinidos a tareas. | |
| Urgencia | 3 | |
| Esfuerzo | 2 | |
| **Pre-condiciones** | Acceso a la tarea. | |
| **Flujo Normal** | Usuario | Sistema |
| Selecciona color. |  |
|  | Persiste etiqueta y actualiza vistas. |
| **Post-condiciones** | Color aplicado para filtros/visualización. | |
| **Excepciones** | E1 Paleta no disponible: Mensaje y se mantiene estado. | |



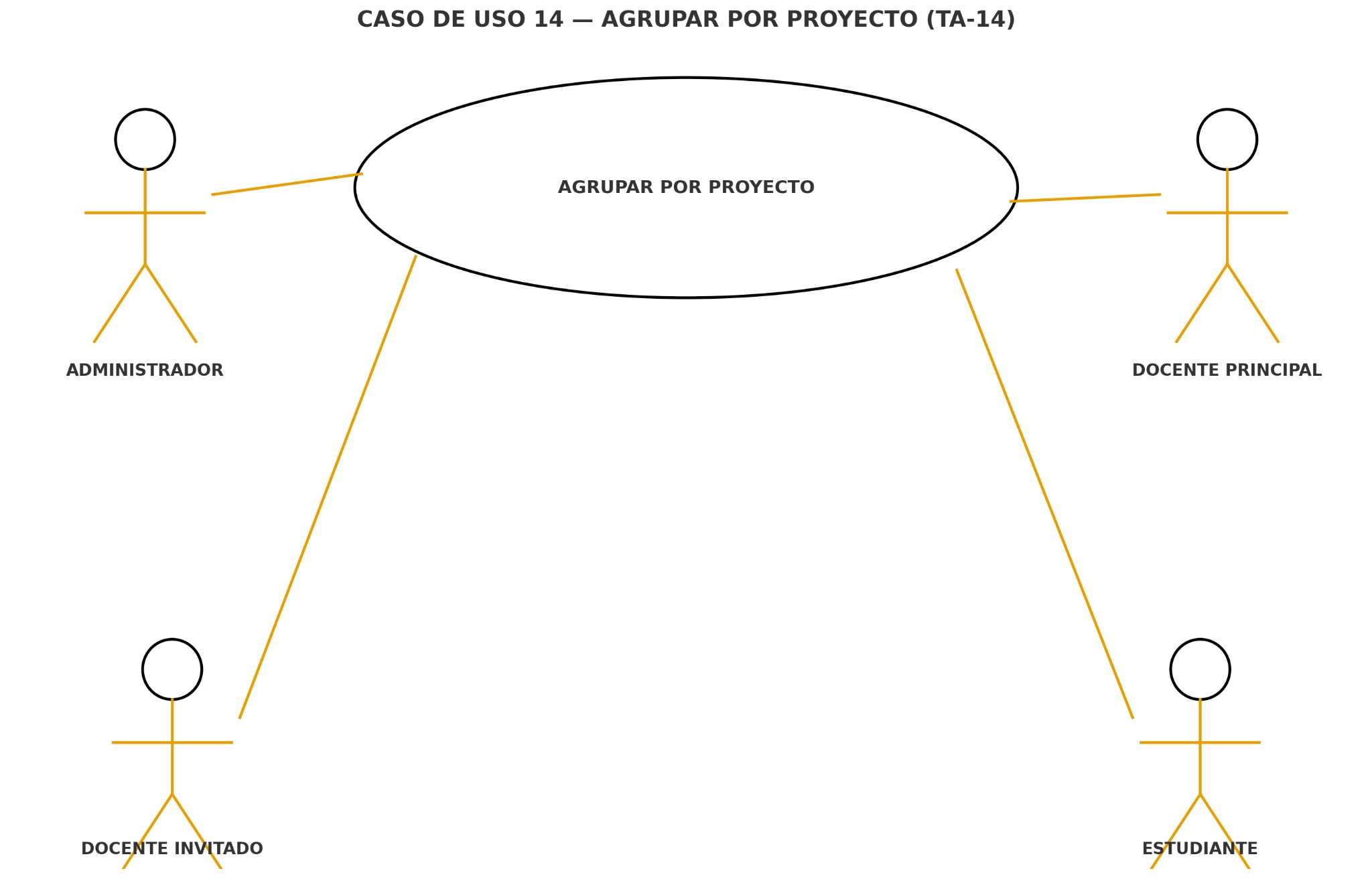
## Etiquetar por Palabras Clave

|  |  |  |
| --- | --- | --- |
|  | TA-13 | |
| **Nombre** | Etiquetar por Palabras Clave | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Añadir etiquetas de texto para clasificación/búsqueda. | |
| Urgencia | 3 | |
| Esfuerzo | 2 | |
| **Pre-condiciones** | Política de longitud y caracteres válidos. | |
| **Flujo Normal** | Usuario | Sistema |
| Escribe etiquetas. |  |
|  | Normaliza, guarda y actualiza índice de búsqueda. |
| **Post-condiciones** | Etiquetas disponibles para filtros y búsqueda. | |
| **Excepciones** | E1 Etiqueta inválida/duplicada: Se notifica y evita duplicidad. | |



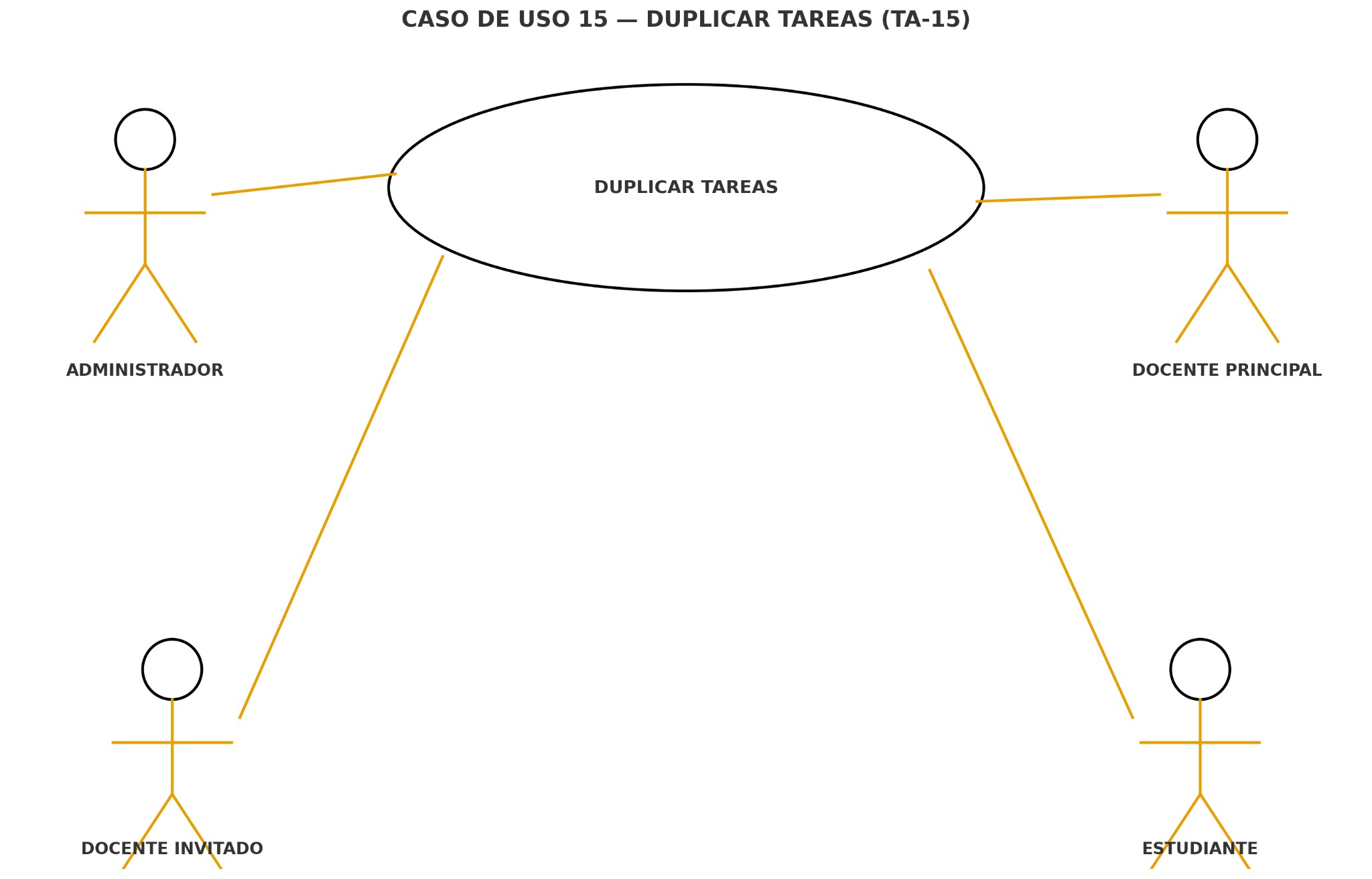
## Agrupar por Proyecto

|  |  |  |
| --- | --- | --- |
|  | TA-14 | |
| **Nombre** | Agrupar por Proyecto | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Organizar y visualizar tareas por proyecto. | |
| Urgencia | 3 | |
| Esfuerzo | 2 | |
| **Pre-condiciones** | Tareas asociadas a proyectos. | |
| **Flujo Normal** | Usuario | Sistema |
| Activa agrupación por proyecto. |  |
|  | Reagrupa y muestra el panel.. |
| **Post-condiciones** | Vista agrupada actualizada. | |
| **Excepciones** | E1 Sin proyecto: Se ofrece asociar a uno. | |



## Duplicar Tareas

|  |  |  |
| --- | --- | --- |
|  | TA-15 | |
| **Nombre** | Duplicar Tareas | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Crear una copia de una tarea con campos seleccionados. | |
| Urgencia | 3 | |
| Esfuerzo | 2 | |
| **Pre-condiciones** | Tarea de origen accesible. | |
| **Flujo Normal** | Usuario | Sistema |
| Elige “Duplicar”. |  |
|  | Ofrece campos a replicar. |
| Confirma. |  |
|  | Crea copia y la muestra. |
| **Post-condiciones** | Nueva tarea con nuevo ID. | |
| **Excepciones** | E1 Campos obligatorios faltantes: Solicitar completarlos. | |



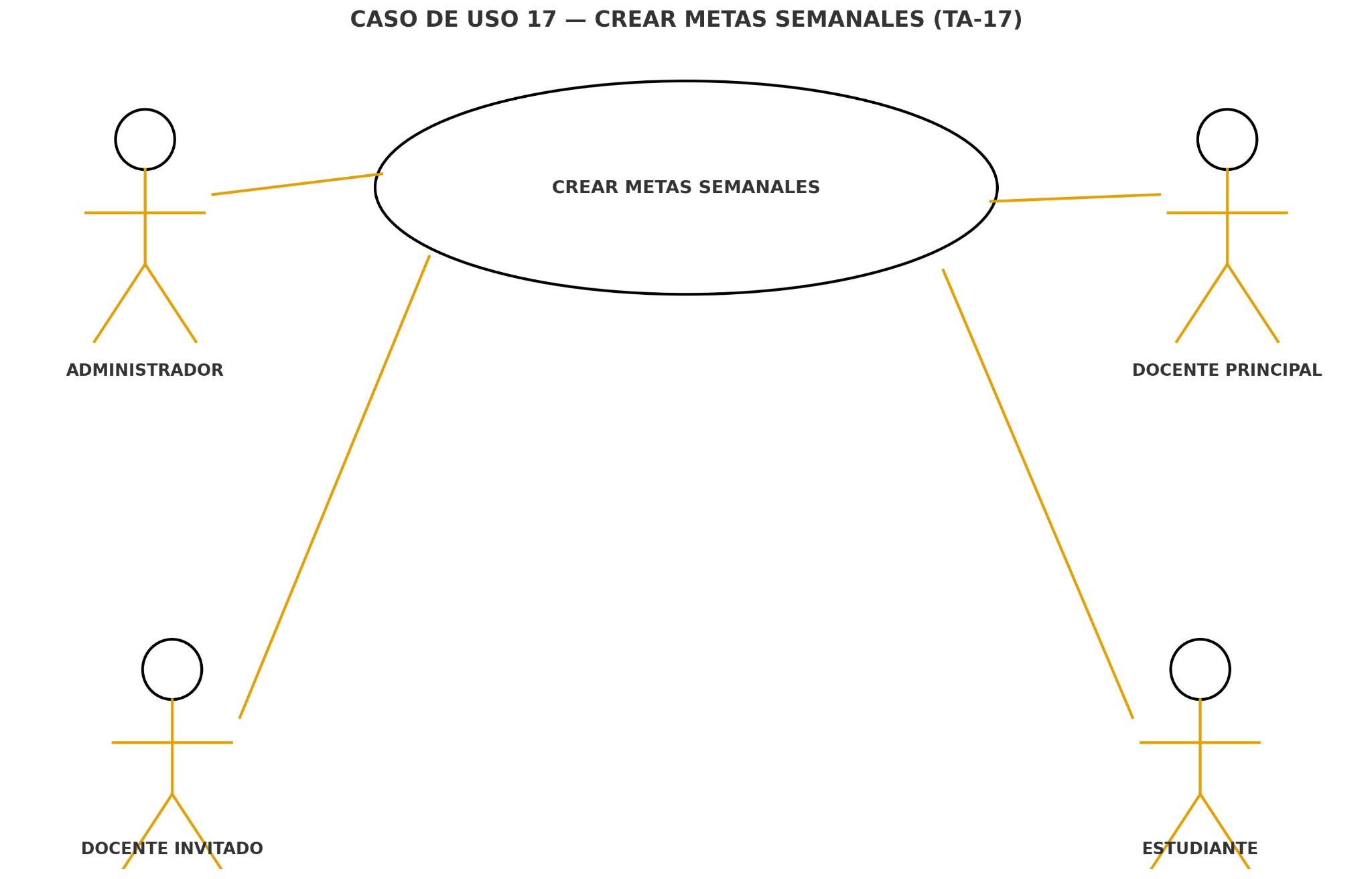
## Guardar Cambios de Tarea

|  |  |  |
| --- | --- | --- |
|  | TA-16 | |
| **Nombre** | Guardar Cambios de Tarea | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Persistir ediciones con control de integridad. | |
| Urgencia | 5 | |
| Esfuerzo | 2 | |
| **Pre-condiciones** | Permisos de edición. | |
| **Flujo Normal** | Usuario | Sistema |
| Edita campos. |  |
|  | Válida. |
| Guarda. |  |
|  | Persiste y registra auditoría. |
| **Post-condiciones** | Tarea actualizada y trazada. | |
| **Excepciones** | E1 Conflicto de edición: Se ofrece reintentar tras recarga. | |



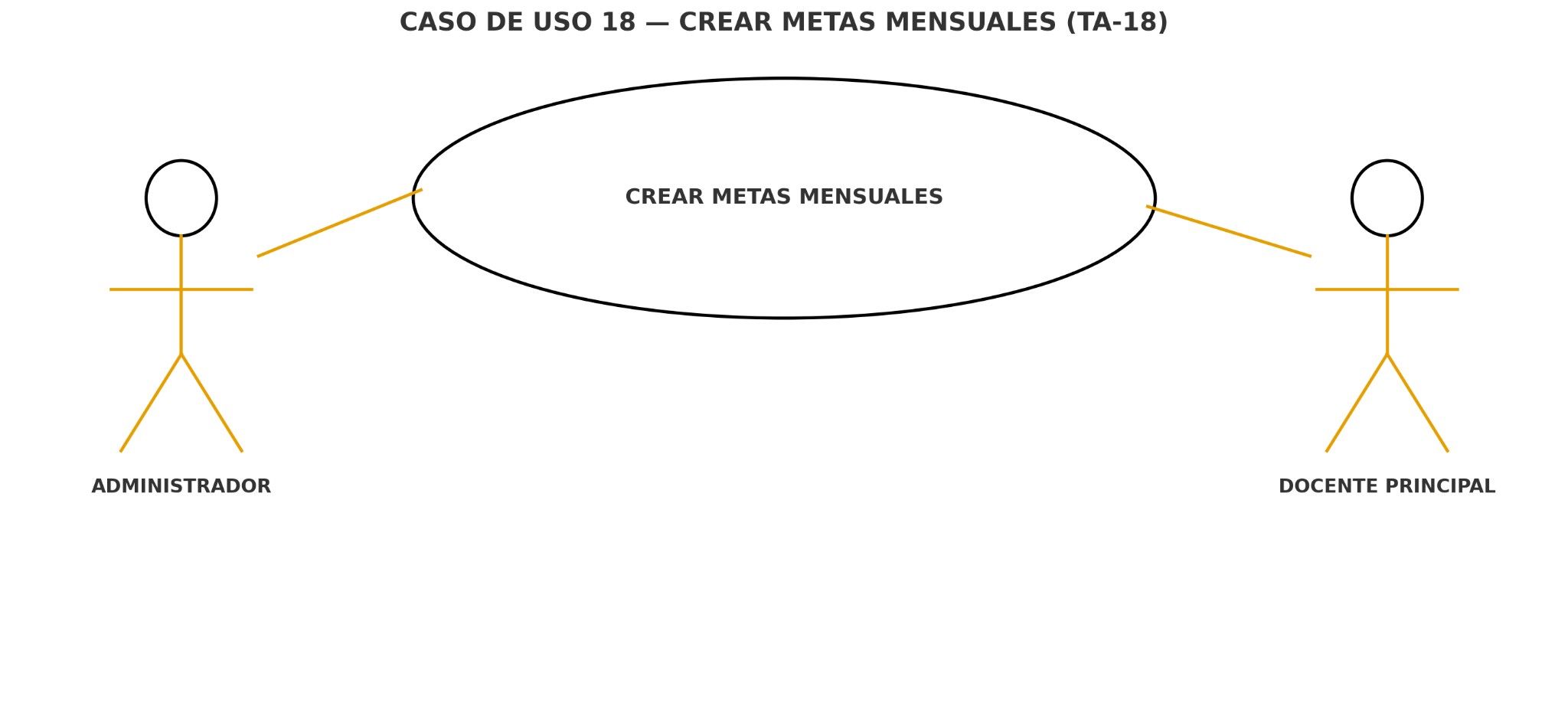
## Crear Metas Semanales

|  |  |  |
| --- | --- | --- |
|  | TA-17 | |
| **Nombre** | Crear Metas Semanales | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Definir objetivos semanales vinculados a tareas. | |
| Urgencia | 3 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Semana seleccionada.  Acceso al proyecto/tarea. | |
| **Flujo Normal** | Usuario | Sistema |
| Crea metas y vincula tareas. |  |
|  | Guarda y calcula el progreso. |
| **Post-condiciones** | Meta registrada con % cumplimiento. | |
| **Excepciones** | E1 Meta duplicada: Se informa y evita duplicidad. | |



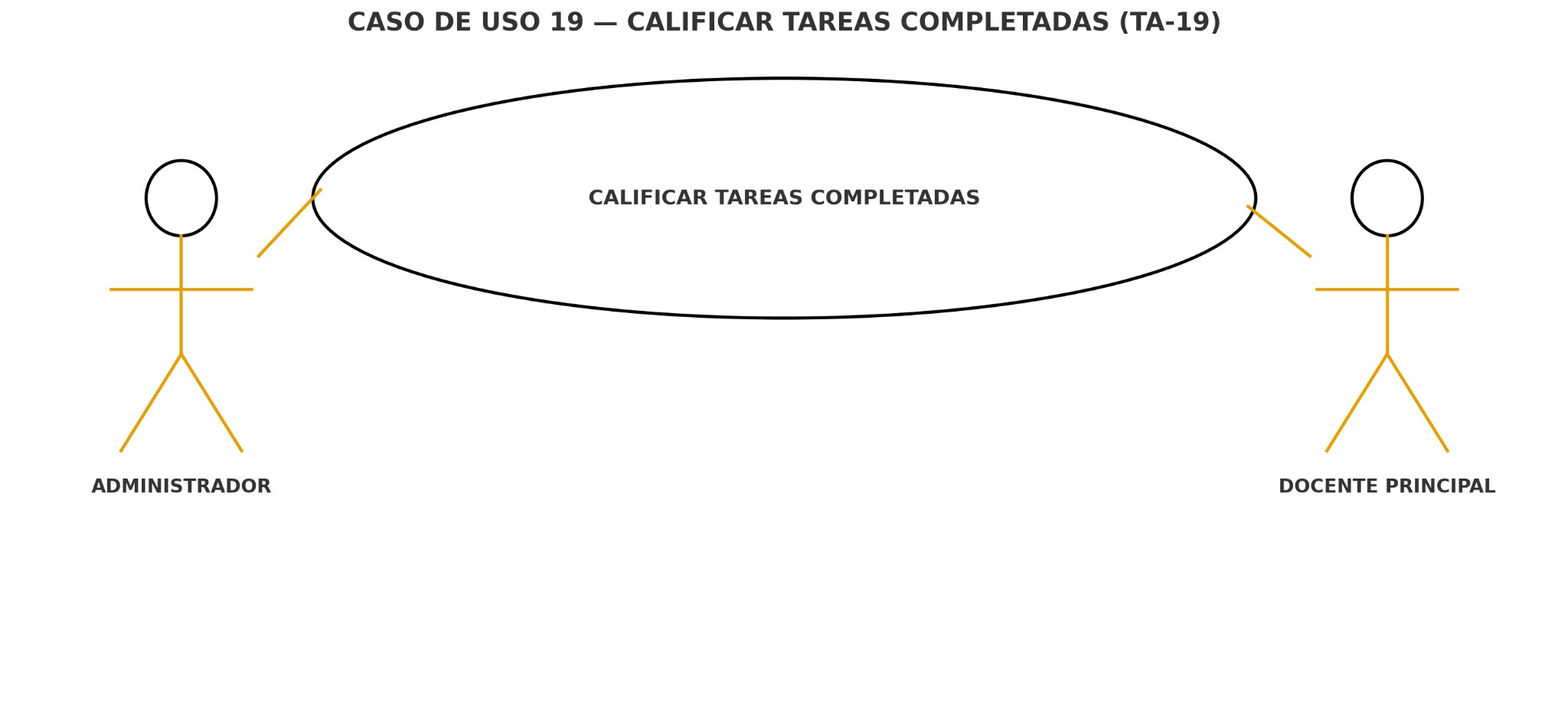
## Crear Metas Mensuales

|  |  |  |
| --- | --- | --- |
|  | TA-18 | |
| **Nombre** | Crear Metas Mensuales | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Planificar metas mensuales y submetas. | |
| Urgencia | 3 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Mes objetivo definido. | |
| **Flujo Normal** | Usuario | Sistema |
| Define metas y submetas. |  |
|  | Guarda y presenta el tablero mensual. |
| **Post-condiciones** | Meta mensual activa con seguimiento. | |
| **Excepciones** | E1 Fechas incoherentes: Validación y corrección. | |



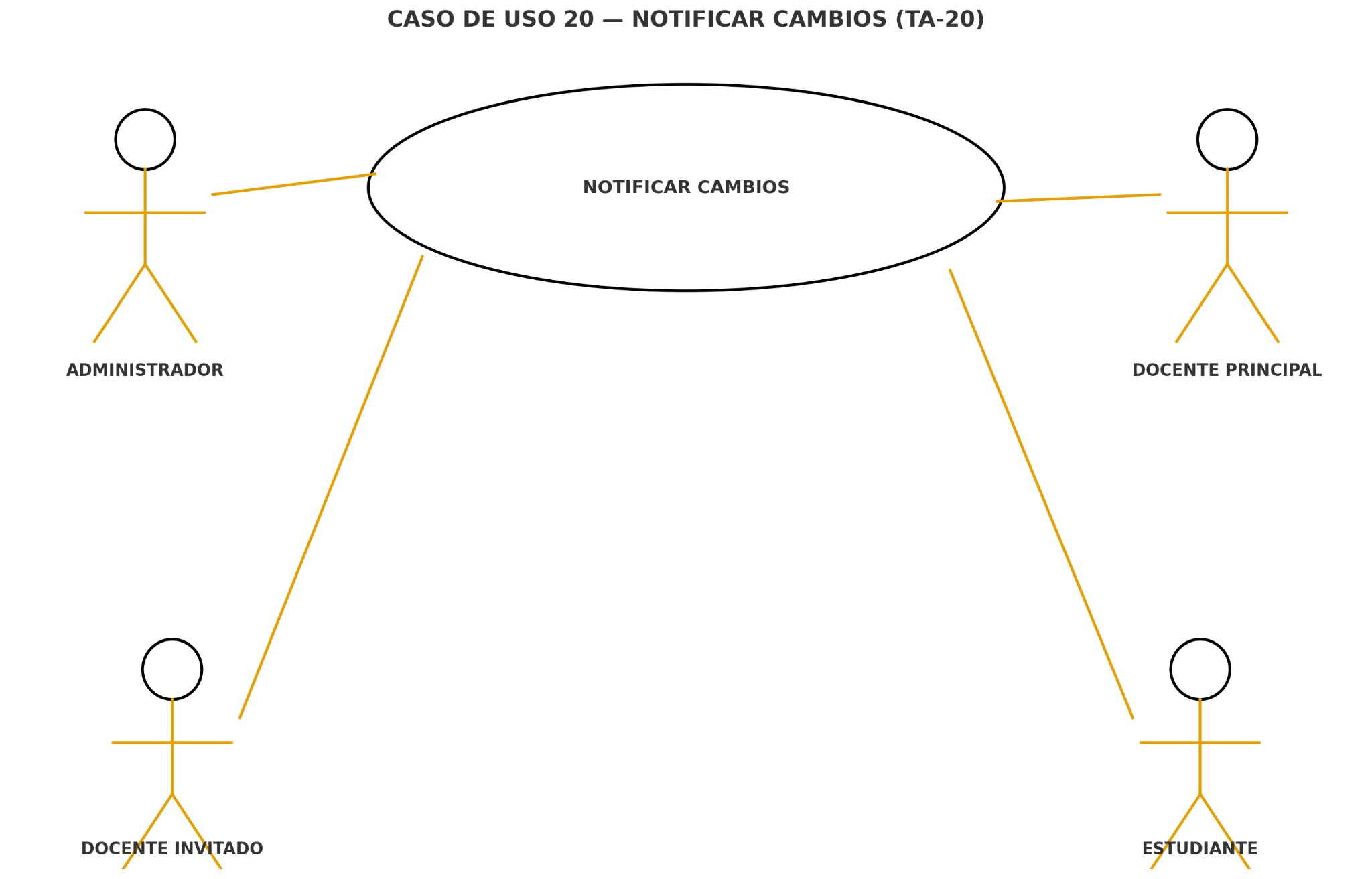
## Calificar Tareas Completadas

|  |  |  |
| --- | --- | --- |
|  | TA-19 | |
| **Nombre** | Calificar Tareas Completadas | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Asignar calificación y feedback a tareas finalizadas. | |
| Urgencia | 4 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Tarea en estado “completada”.  Permisos de calificación. | |
| **Flujo Normal** | Usuario | Sistema |
| Abre el panel de calificación. |  |
|  | Muestra rúbrica/escala. |
| Ingresa nota y feedback. |  |
|  | Persiste y notifica al estudiante. |
| **Post-condiciones** | Calificación visible en el historial. | |
| **Excepciones** | E1 Tarea sin finalizar: Se bloquea calificación. | |



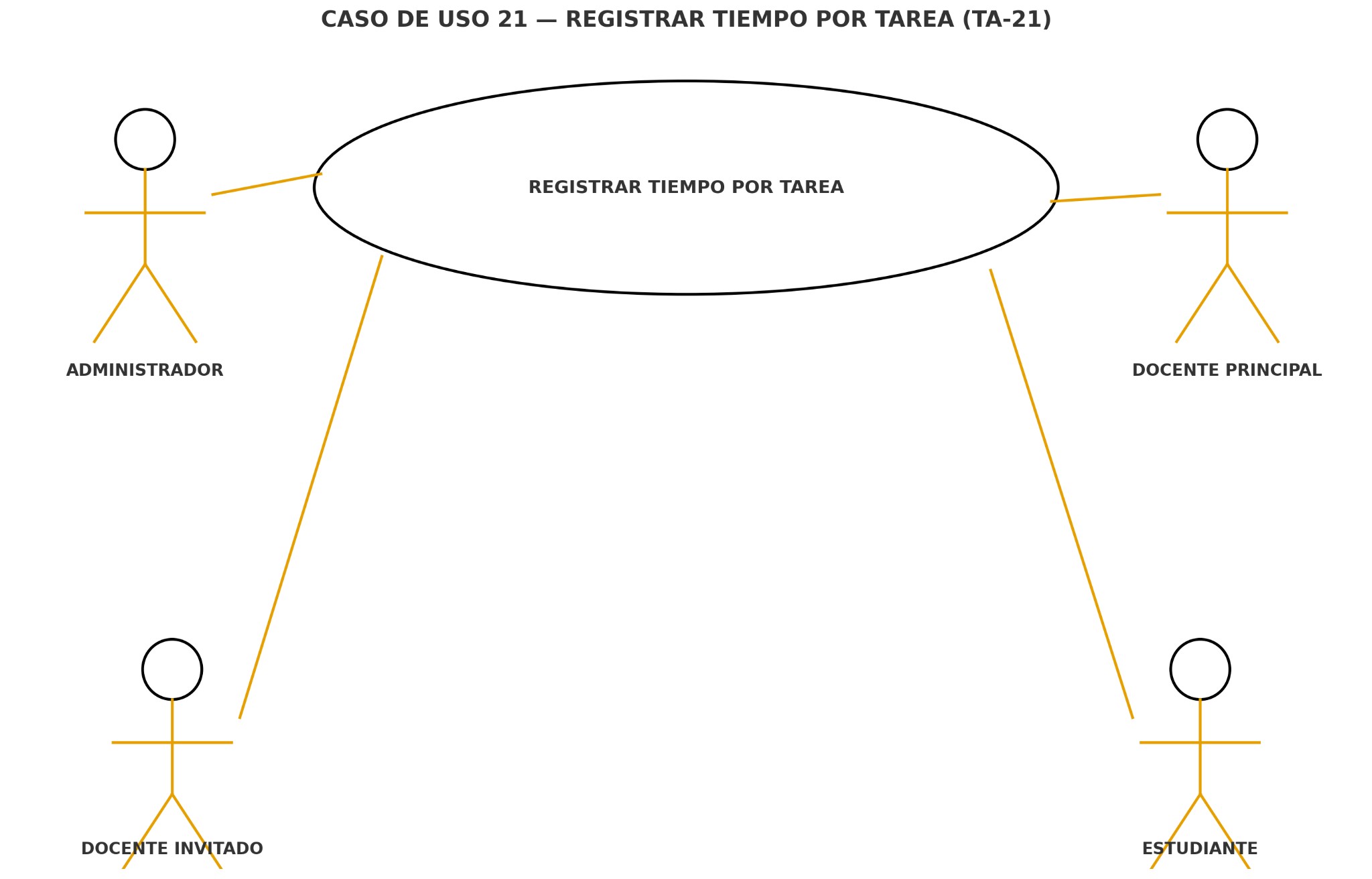
## Notificar Cambios

|  |  |  |
| --- | --- | --- |
|  | TA-20 | |
| **Nombre** | Notificar Cambios | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Enviar alertas por eventos relevantes. | |
| Urgencia | 4 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Preferencias de notificación definidas. | |
| **Flujo Normal** | Usuario | Sistema |
| Detecta evento (asignación, fecha modificada, mención, vencimiento). |  |
|  | Envía notificación según canal preferido. |
| **Post-condiciones** | Evento comunicado y registrado. | |
| **Excepciones** | E1 Opt-out del usuario: No se envía. | |



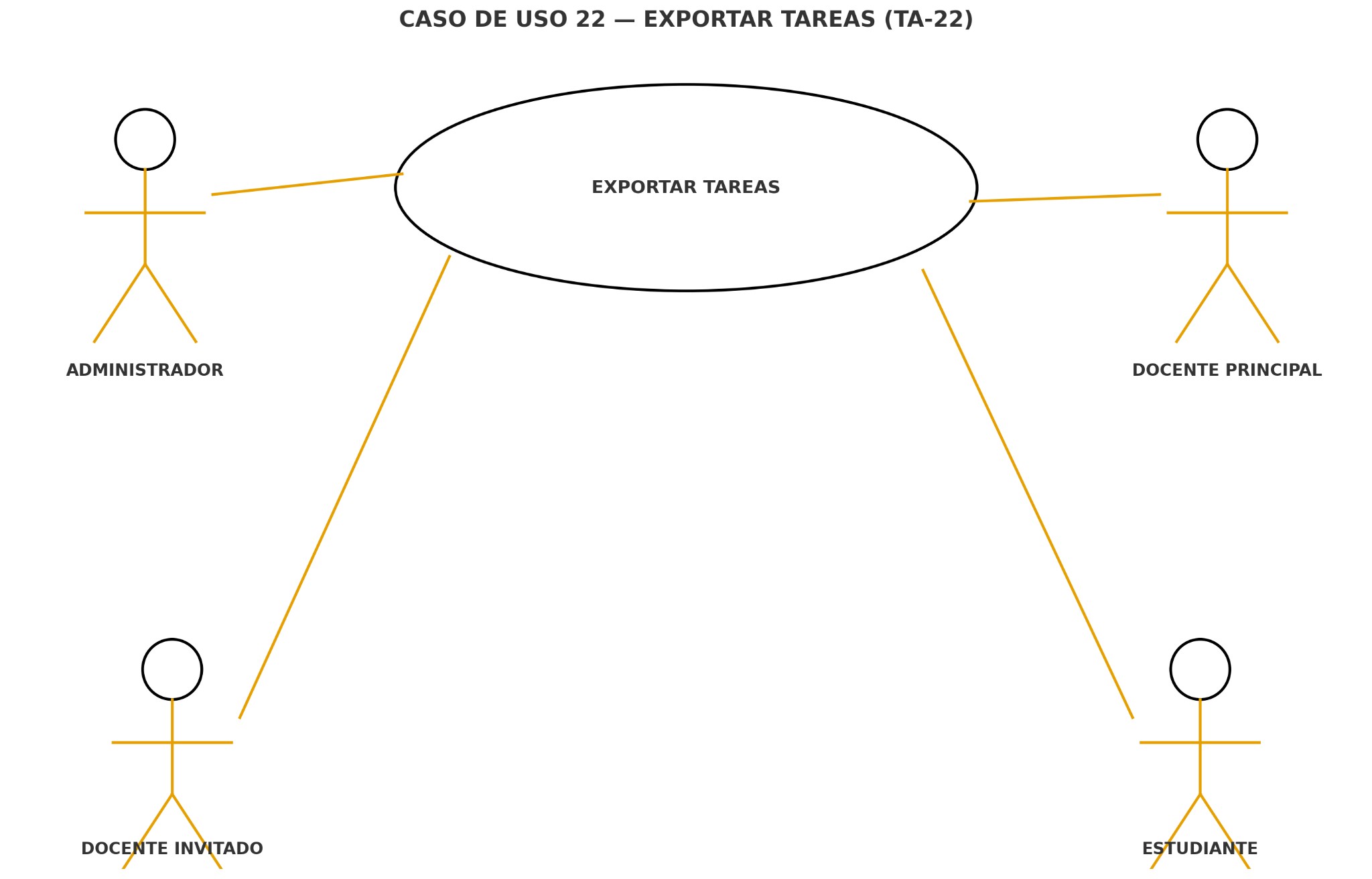
## Registrar Tiempo por Tarea

|  |  |  |
| --- | --- | --- |
|  | TA-21 | |
| **Nombre** | Registrar Tiempo por Tarea | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Registrar tiempo invertido por usuario. | |
| Urgencia | 3 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Acceso a la tarea. | |
| **Flujo Normal** | Usuario | Sistema |
| Inicia/ingresa tiempo. |  |
|  | Valida y guarda. |
| Detiene o edita entrada. |  |
|  | Actualiza totales. |
| **Post-condiciones** | Tiempo registrado por usuario y por tarea. | |
| **Excepciones** | E1 Duración inválida: Bloqueo y mensaje. | |



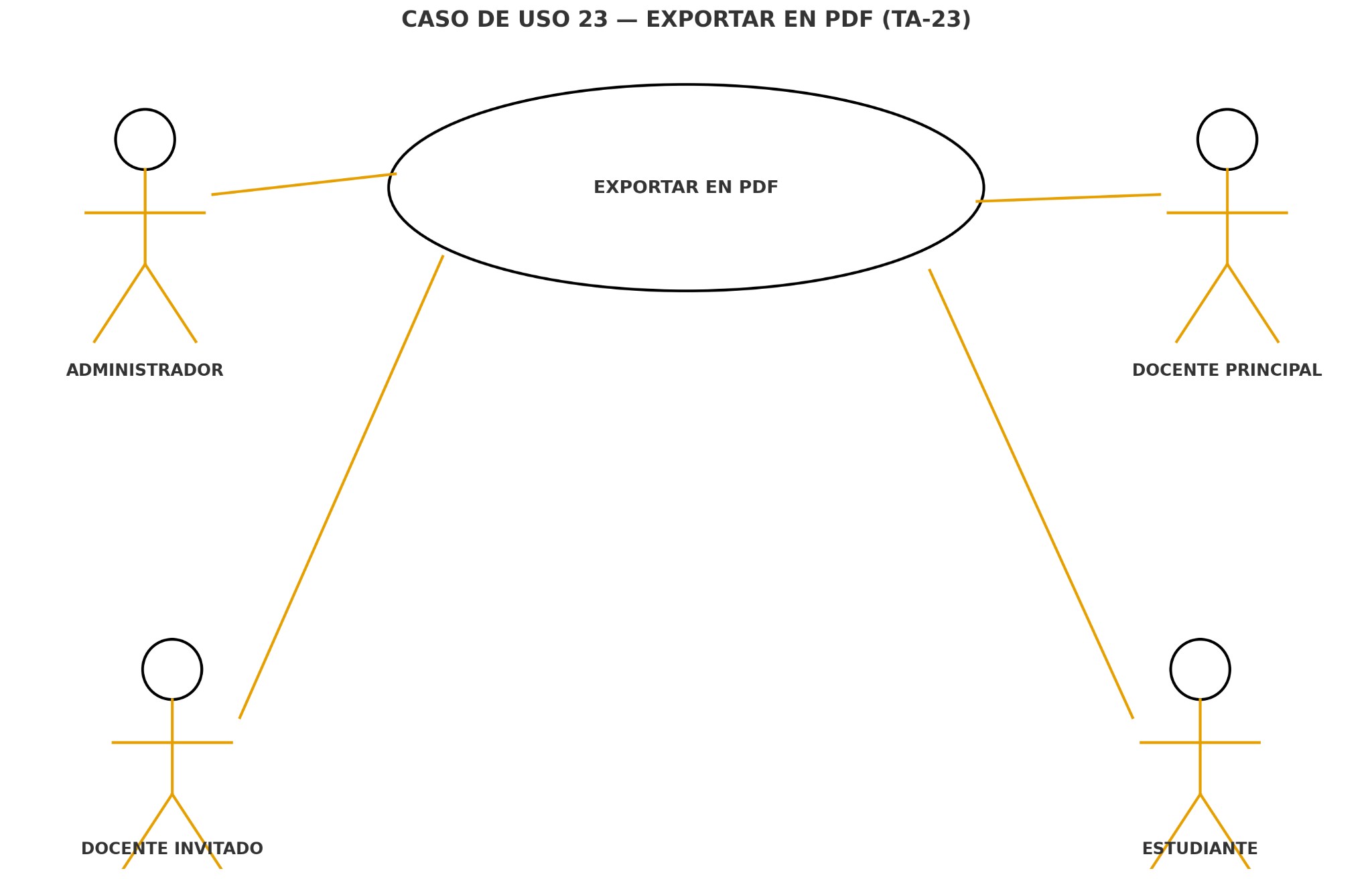
## Exportar Tareas

|  |  |  |
| --- | --- | --- |
|  | TA-22 | |
| **Nombre** | Exportar Tareas | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Exportar conjuntos filtrados de tareas. | |
| Urgencia | 3 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Filtros/selección definidos. | |
| **Flujo Normal** | Usuario | Sistema |
| Define filtros y formato. |  |
|  | Genera archivo y ofrece descarga. |
| **Post-condiciones** | Archivo disponible al usuario. | |
| **Excepciones** | E1 Límite de filas: Solicita reducir rango o segmentar. | |



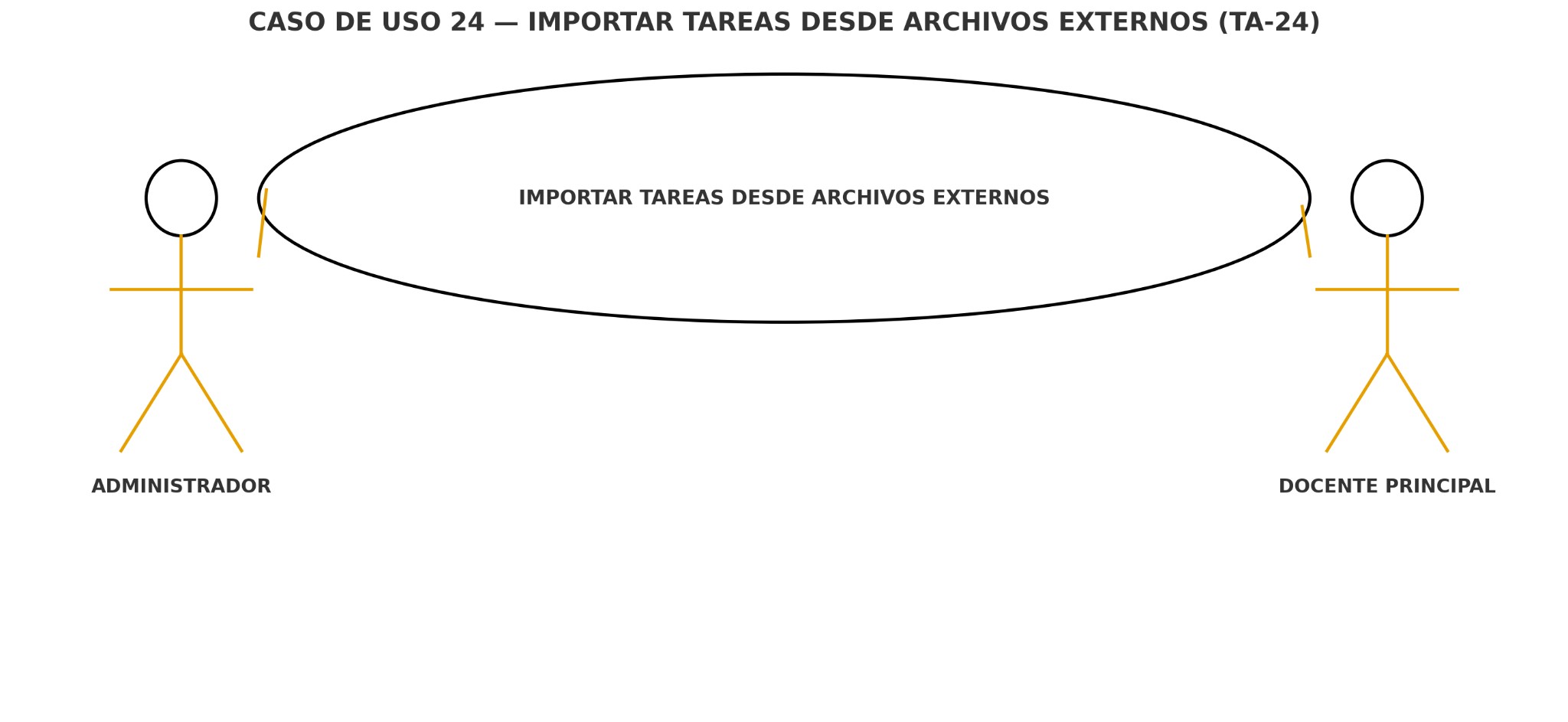
## Exportar en PDF

|  |  |  |
| --- | --- | --- |
|  | TA-23 | |
| **Nombre** | Exportar en PDF | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Generar reportes PDF con formato. | |
| Urgencia | 3 | |
| Esfuerzo | 3 | |
| **Pre-condiciones** | Datos filtrados. | |
| **Flujo Normal** | Usuario | Sistema |
| Elige “PDF”. |  |
|  | Renderiza plantilla y entrega archivo. |
| **Post-condiciones** | PDF descargado/almacenado. | |
| **Excepciones** | E1 Plantilla no disponible: Muestra error y reintento. | |



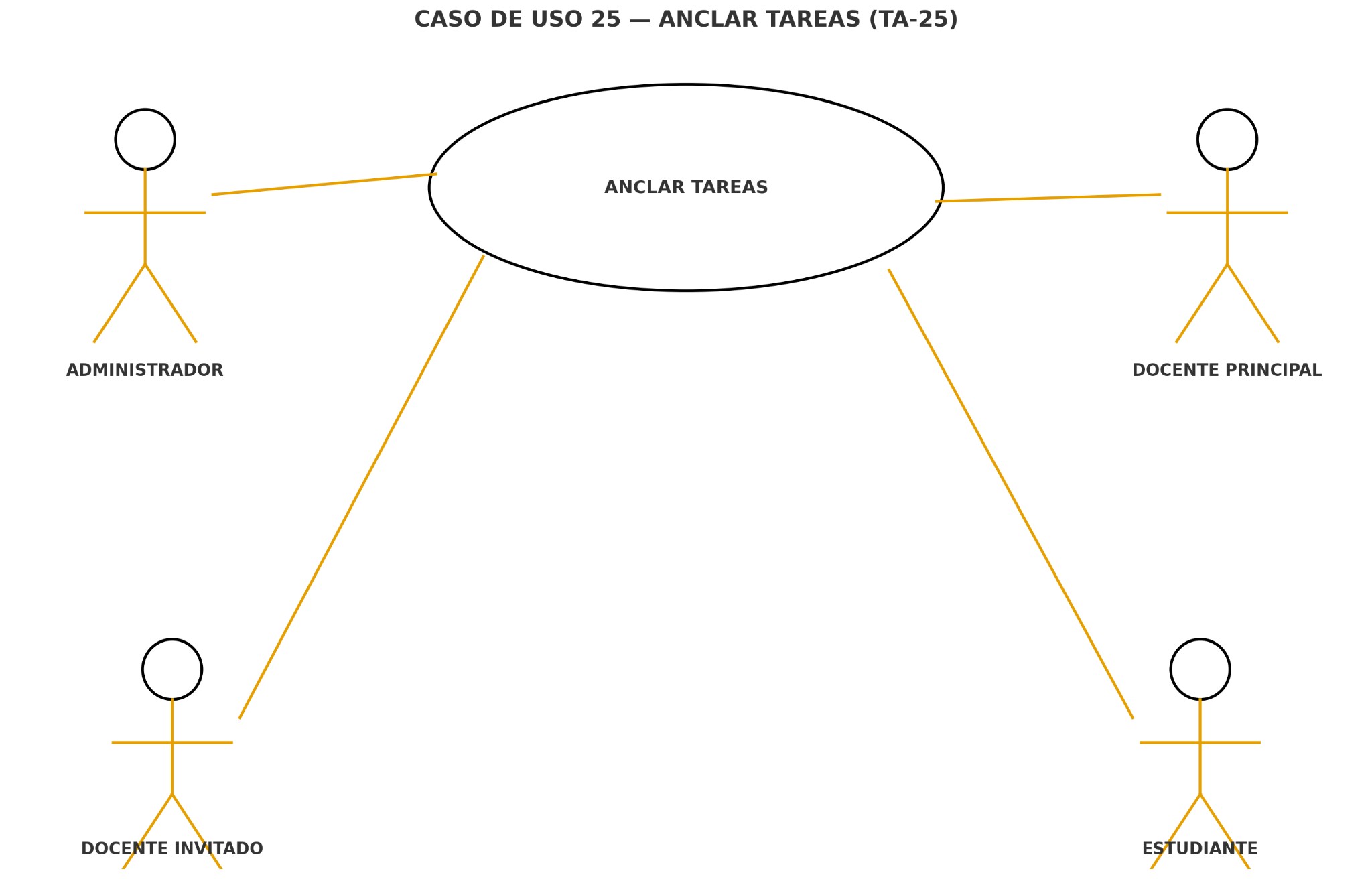
**Importar Tareas desde Archivos Externos**

|  |  |  |
| --- | --- | --- |
| **ID:** | TA-24 | |
| **Nombre** | Crear tarea | |
| **Actores** | Administrador, Docente Principal | |
| **Objetivo** | Incorporar tareas vía CSV/Excel con validación previa. | |
| Urgencia | 3 | |
| Esfuerzo | 4 | |
| **Pre-condiciones** | Archivo en formato soportado.  Mapeo de columnas válido. | |
| **Flujo Normal** | Usuario | Sistema |
| Sube archivo. |  |
|  | Prevalida y muestra previsualización. |
| Ajusta mapeo y confirma importación. |  |
|  | Inserta registros y resume resultados. |
| **Post-condiciones** | Tareas importadas y errores reportados. | |
| **Excepciones** | E1 Formato inválido: Rechazo con detalle de columnas. | |

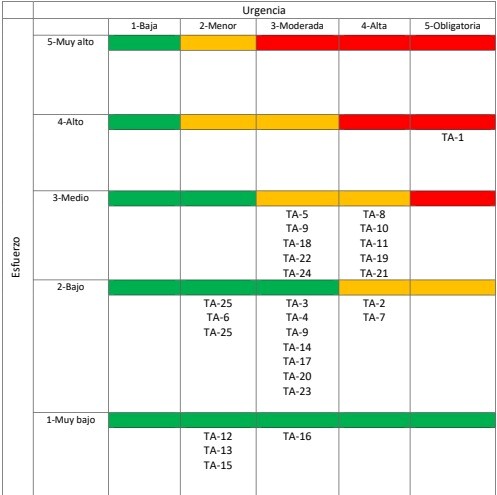


## Anclar Tareas

|  |  |  |
| --- | --- | --- |
| **ID:** | TA-25 | |
| **Nombre** | Anclar Tareas | |
| **Actores** | Administrador, Docente Principal, Docente invitado, Estudiante. | |
| **Objetivo** | Fijar tareas prioritarias en la parte superior de vistas personales. | |
| Urgencia | 3 | |
| Esfuerzo | 2 | |
| **Pre-condiciones** | Acceso a la tarea. | |
| **Flujo Normal** | Usuario | Sistema |
| Pulsa “Anclar”. |  |
|  | Marca la tarea y la posiciona en la sección de anclados. |
| **Post-condiciones** | Tarea visible en “Ancladas” del usuario. | |
| **Excepciones** | E1 Límite de anclados alcanzado: Mensaje y no se ancla. | |



# Matriz de Prioridad



## Prioridad de Requerimientos

### 4. Requisitos No Funcionales

#### 4.1 Seguridad (RNF-SEC)

Se definió que Task Assistant debía garantizar que solo las personas autorizadas puedan acceder a la plataforma y a la información. Para ello, se implementó un esquema de autenticación y autorización basado en roles (RBAC) con tokens JWT, de manera que cada usuario inicia sesión, obtiene un token con su rol y este se verifica en cada acción; estos tokens tienen fecha de expiración y pueden renovarse de forma segura. Toda la comunicación entre el navegador y el servidor se realiza por HTTPS (TLS 1.2 o superior), de modo que los datos viajen cifrados. La información sensible, como los datos personales y las credenciales, se protege con cifrado fuerte en reposo (AES-256) y las contraseñas se almacenan mediante algoritmos seguros como Argon2id o bcrypt con salt, lo que evita que puedan recuperarse en texto plano. Además, se configuró una gestión de sesiones que cierra automáticamente la sesión tras al menos 30 minutos de inactividad y revoca los tokens cuando el usuario cierra sesión. Para reducir riesgos de abuso o ataque, se incluyeron mecanismos de limitación de peticiones (rate limiting), protección frente a ataques CSRF, validación estricta de toda la información que ingresa el usuario y cabeceras de seguridad como CSP, HSTS y X-Content-Type-Options. Finalmente, se estableció un sistema de auditoría que registra los eventos críticos (creación, edición, eliminación y recuperación de tareas, calificaciones, exportaciones y autenticación), conservando estos registros durante al menos 365 días e incluyendo quién realizó la acción, desde qué IP y en qué momento.

#### 4.2 Desempeño (RNF-PERF)

El objetivo fue que la aplicación se percibiera fluida incluso con un uso intensivo. Se definió que las operaciones más habituales de creación, consulta, actualización y eliminación de información (CRUD) debían completarse, en el 95 % de los casos, en menos de 300 milisegundos y, en el 99 %, en menos de 700 milisegundos. De manera más general, acciones como crear, editar, eliminar, recuperar tareas, añadir comentarios o menciones no debían tardar más de 2 segundos para el usuario. El tablero principal, aun cuando muestra más de 100 tareas activas, se diseñó para quedar listo para la interacción en un máximo de 5 segundos. Para procesos más pesados, como exportar datos a PDF o Excel, importar archivos o realizar múltiples sincronizaciones, se estableció un tiempo objetivo de hasta 10 segundos en condiciones óptimas. Además, se planteó que el sistema pudiera soportar hasta 10 000 usuarios concurrentes, permitiendo como máximo una degradación de rendimiento del 20 %, de modo que el servicio siga siendo utilizable bajo alta demanda.

#### 4.3 Disponibilidad y Continuidad (RNF-DISP)

El requisito RNF-DISP-01 fijó un tiempo de actividad (uptime) mensual mínimo del 99,5 %, lo que implica que la plataforma debe estar disponible prácticamente todo el tiempo. El requisito RNF-DISP-02 definió una estrategia de copias de seguridad automáticas diarias, con un objetivo de punto de recuperación (RPO) de hasta 15 minutos es decir, en caso de incidente se podría perder como máximo la información generada en los últimos 15 minutos y un objetivo de tiempo de recuperación (RTO) de hasta 60 minutos para restablecer el servicio. El requisito RNF-DISP-03 estableció que el despliegue debía ser tolerante a fallos, manteniendo al menos dos instancias por servicio detrás de un balanceador de carga, de forma que, si una instancia falla, otra pueda asumir la carga sin interrumpir el acceso de los usuarios.

#### 4.4 Escalabilidad (RNF-ESC)

El requisito RNF-ESC-01 definió que la base de datos debía diseñarse con índices adecuados y preparada para aplicar particionamiento o replicación cuando el volumen de datos creciera, manteniendo las consultas dentro de tiempos aceptables (P95 ≤ 150 ms). El requisito RNF-ESC-02 estableció que el backend debía poder escalar horizontalmente, añadiendo más instancias (contenedores) detrás de un balanceador de carga a medida que creciera la demanda, y apoyarse en colas de procesamiento para tareas pesadas que no sea necesario resolver en tiempo real. El requisito RNF-ESC-03 complementó lo anterior con el uso de mecanismos de caché para lecturas frecuentes y carga diferida (lazy loading) en listados muy largos, evitando bloquear al usuario mientras se carga toda la información.

#### 4.5 Usabilidad y Accesibilidad (RNF-USA)

En usabilidad y accesibilidad, el requisito RNF-USA-01 indicó que la interfaz debía ofrecer una navegación clara, con menús y rutas comprensibles, además de atajos de teclado y estados de foco visibles para orientar al usuario. El requisito RNF-USA-02 estableció un proceso de incorporación (onboarding) con una guía interactiva contextual y ayuda integrada dentro de la propia aplicación, para que los nuevos usuarios comprendan rápidamente cómo usar las principales funciones. El requisito RNF-USA-03 señaló que el sistema debía ofrecer mensajes de confirmación, error o éxito en menos de 500 milisegundos tras la acción del usuario, de modo que siempre reciba una retroalimentación rápida sobre lo que ha ocurrido. Finalmente, el requisito RNF-USA-04 recogió el compromiso con el cumplimiento de las pautas de accesibilidad WCAG 2.1 en nivel AA, incluyendo soporte para lectores de pantalla, uso de atributos ARIA y la posibilidad de navegar completamente mediante teclado.

#### 4.6 Compatibilidad y Portabilidad (RNF-COMP)

El requisito RNF-COMP-01 estableció que Task Assistant debía funcionar como una aplicación web progresiva (PWA) en los principales sistemas operativos de escritorio y móviles (Windows, macOS, Android e iOS). El requisito RNF-COMP-02 definió que la herramienta debía ser plenamente funcional en las dos últimas versiones de los navegadores más utilizados (Chrome, Edge, Firefox y Safari), garantizando así una experiencia consistente para la mayoría de usuarios. El requisito RNF-COMP-03 añadió la necesidad de ofrecer soporte en español (es-ES) y dejar preparada la base técnica para la internacionalización (i18n), de manera que sea sencillo incorporar otros idiomas en el futuro.

#### 4.7 Cumplimiento Normativo y Privacidad (RNF-LEG)

El requisito RNF-LEG-01 estableció que el tratamiento de datos personales debía ajustarse a principios como la minimización (recoger sólo lo estrictamente necesario), el consentimiento informado y el respeto a los derechos de acceso, rectificación, cancelación y oposición (derechos ARCO) de las personas usuarias. El requisito RNF-LEG-02 definió que debían existir políticas claras de retención y borrado de la información, alineadas con la normativa local aplicable, de forma que los datos no se conserven más tiempo del necesario y puedan eliminarse de forma segura cuando corresponda.

#### 4.8 Mantenibilidad y Calidad (RNF-MANT)

El requisito RNF-MANT-01 dispuso que el código debía contar con una cobertura de pruebas de al menos el 80 % en los módulos críticos, complementada con revisiones por pares y un flujo de integración y despliegue continuo (CI/CD), lo que facilita detectar errores antes de llegar a producción. El requisito RNF-MANT-02 indicó que tanto la API como los principales casos de uso debían estar documentados de manera clara, utilizando versionado semántico para identificar cambios y mantener la trazabilidad de las versiones. El requisito RNF-MANT-03 señaló que los parámetros más importantes del sistema como límites, umbrales o configuraciones de notificaciones debían poder gestionarse desde el entorno o desde un módulo de administración, evitando depender de cambios directos en el código para ajustar el comportamiento de la herramienta.

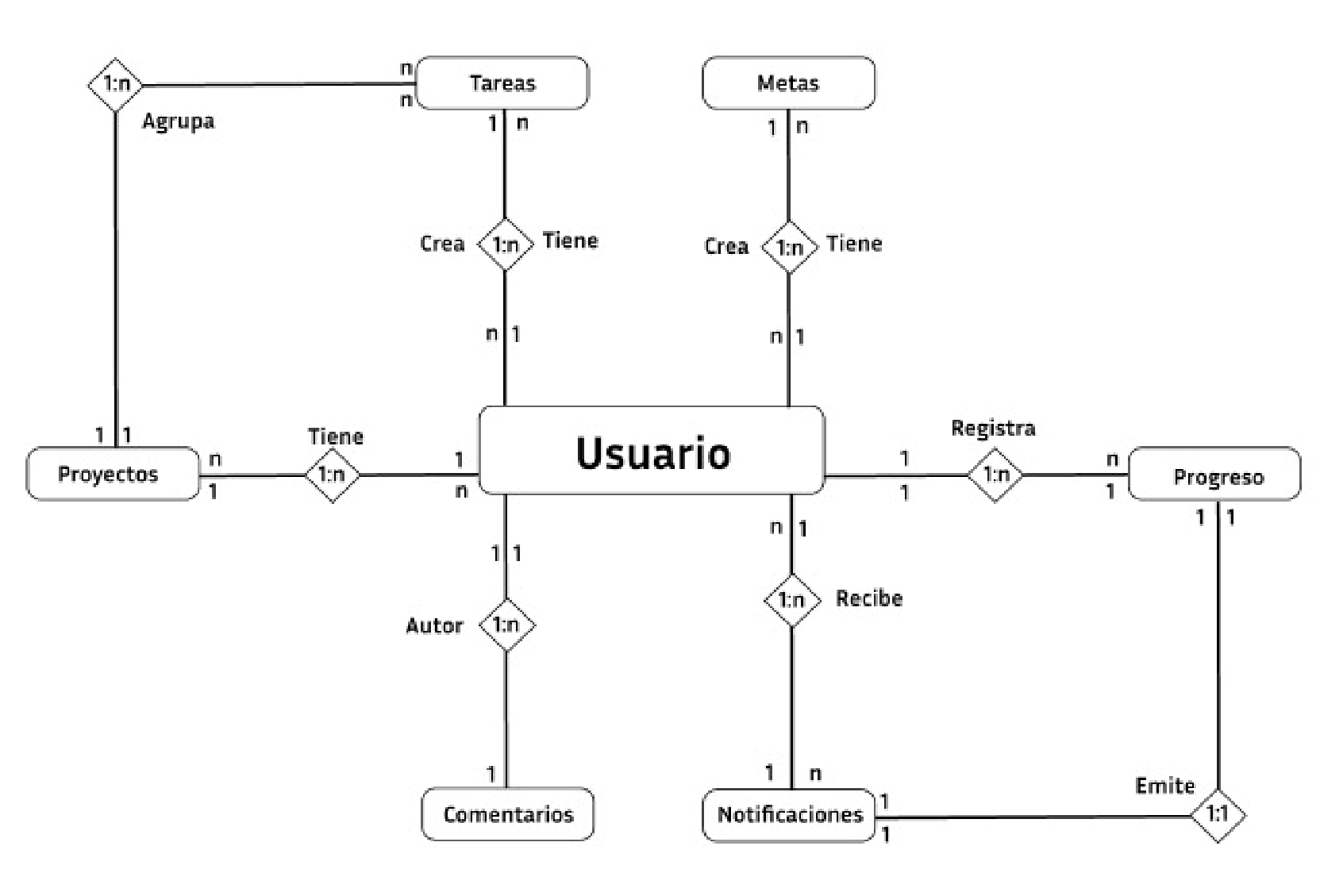
#### 4.9 Observabilidad (RNF-OBS)

Finalmente, en observabilidad, el requisito RNF-OBS-01 indicó que el sistema debía contar con métricas y trazas que permitieran monitorear las latencias (P95 y P99), la tasa de errores y el volumen de peticiones, así como disponer de trazas distribuidas (por ejemplo, con OpenTelemetry o una herramienta equivalente) para seguir el recorrido de una solicitud a través de los distintos servicios. El requisito RNF-OBS-02 complementó esto con la definición de umbrales y alertas en tiempo real ante degradaciones relevantes, como aumentos de latencia, incremento de errores, problemas de espacio en disco o acumulación en las colas de trabajo, de modo que el equipo técnico pueda actuar de forma oportuna cuando algo empieza a fallar.

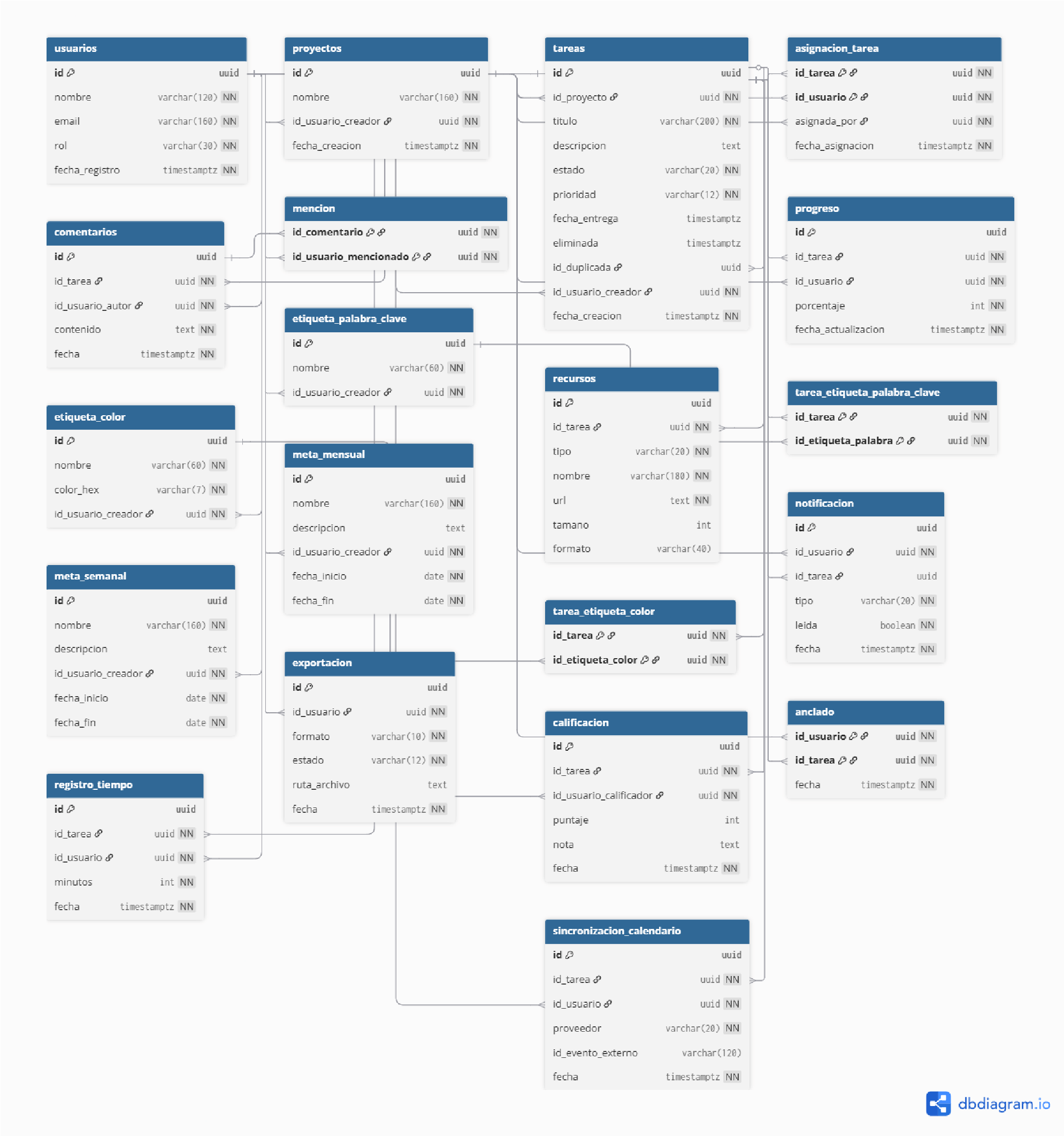
## 5.Modelado E/R

### Diagrama de Entidad-Relación

Para el diseño del modelo de datos de TaskAssistant se partió de la identificación de las entidades principales que intervienen en la gestión de tareas colaborativas: usuarios, tareas, proyectos, recursos, comentarios y metas. A partir de estas entidades se definieron las relaciones necesarias para que cada usuario pudiera participar en uno o varios proyectos, recibir tareas específicas, asociarles recursos (documentos, enlaces, materiales de apoyo), intercambiar comentarios en torno a esas tareas y vincular su trabajo con metas individuales o de equipo. De este modo, el modelo permitió organizar el trabajo en proyectos, asignar responsabilidades de forma clara y hacer seguimiento en tiempo casi real al avance de las tareas distribuidas entre los distintos miembros del equipo, manteniendo siempre el contexto de quién hace qué, con qué materiales y con qué objetivo.



## Diagrama Relacional



## Script de modelo relacional

TaskAssistant - Modelo relacional (dbdiagram.io)

Table Usuario { id varchar [pk] nombre varchar email varchar [unique] rol varchar // administrador, docente\_principal, docente\_invitado, estudiante fechaRegistro datetime

}

Table Proyecto { id varchar [pk] nombre varchar idUsuarioCreador varchar [ref: > Usuario.id] fechaCreacion datetime

}

Table Tarea { id varchar [pk] idProyecto varchar [ref: > Proyecto.id] titulo varchar descripcion text estado varchar // nueva, en\_progreso, completada, archivada prioridad varchar // baja, media, alta, urgente fechaEntrega datetime eliminada datetime // para recuperación (soft delete) idDuplicada varchar [ref: > Tarea.id] idUsuarioCreador varchar [ref: > Usuario.id] fechaCreacion datetime

}

Table AsignacionTarea { idTarea varchar [ref: > Tarea.id, pk] idUsuario varchar [ref: > Usuario.id, pk] asignadaPor varchar [ref: > Usuario.id] fechaAsignacion datetime

}

Comentario { varchar [pk]

idTarea varchar [ref: > Tarea.id] idUsuarioAutor varchar [ref: > Usuario.id] contenido text fecha datetime

}

Table Mencion { idComentario varchar [ref: > Comentario.id, pk] idUsuarioMencionado varchar [ref: > Usuario.id, pk]

}

Table Recurso { id varchar [pk] idTarea varchar [ref: > Tarea.id] tipo varchar // pdf, img, video, link, otro nombre varchar

url text tamano int formato varchar

}

Table Progreso { id varchar [pk] idTarea varchar [ref: > Tarea.id] idUsuario varchar [ref: > Usuario.id] porcentaje int // 0 a 100 fechaActualizacion datetime

}

Table RegistroTiempo { id varchar [pk] idTarea varchar [ref: > Tarea.id] idUsuario varchar [ref: > Usuario.id] minutos int fecha datetime

}

EtiquetaColor { varchar [pk]

nombre varchar colorHex varchar // #RRGGBB idUsuarioCreador varchar [ref: > Usuario.id]

}

Table EtiquetaPalabraClave { id varchar [pk] nombre varchar idUsuarioCreador varchar [ref: > Usuario.id]

}

Table TareaEtiquetaColor { idTarea varchar [ref: > Tarea.id, pk]

idEtiquetaColor varchar [ref: > EtiquetaColor.id, pk]

}

Table TareaEtiquetaPalabraClave { idTarea varchar [ref: > Tarea.id, pk] idEtiquetaPalabra varchar [ref: > EtiquetaPalabraClave.id, pk]

}

Table MetaSemanal { id varchar [pk] nombre varchar descripcion text idUsuarioCreador varchar [ref: > Usuario.id] fechaInicio date fechaFin date

}

Table MetaMensual { id varchar [pk] nombre varchar descripcion text idUsuarioCreador varchar [ref: > Usuario.id] fechaInicio date fechaFin date

}

Calificacion { varchar [pk]

idTarea varchar [ref: > Tarea.id] idUsuarioCalificador varchar [ref: > Usuario.id] puntaje int // 0 a 100 nota text fecha datetime

}

Table Notificacion { id varchar [pk] idUsuario varchar [ref: > Usuario.id] idTarea varchar [ref: > Tarea.id] tipo varchar // mencion, cambio, asignacion, recordatorio leida bool fecha datetime

}

Table Exportacion { id varchar [pk] idUsuario varchar [ref: > Usuario.id] formato varchar // pdf, excel estado varchar // pendiente, completada, fallida rutaArchivo text fecha datetime

}

Table SincronizacionCalendario { id varchar [pk] idTarea varchar [ref: > Tarea.id] idUsuario varchar [ref: > Usuario.id] proveedor varchar // google, outlook, otro idEventoExterno varchar fecha datetime

}

Table Anclado { idUsuario varchar [ref: > Usuario.id, pk] idTarea varchar [ref: > Tarea.id, pk] fecha datetime

}

### Descripción de Entidades y Relaciones

Las entidades y relaciones se derivan de las funcionalidades confirmadas en el documento (gestión integral de tareas; comentarios y @menciones; recursos; progreso y registro de tiempo; organización por proyectos y etiquetas; metas; calificación; notificaciones; sincronización con calendarios; exportes).

**Usuario (PK id):** actor del sistema con rol (Administrador, Docente Principal, Docente Invitado, Estudiante).

Relacionado como creador de Proyecto y Tarea; receptor de Notificación; asignado a Tarea; registra Progreso y Registro Tiempo; crea Etiquetas y Metas.

**Proyecto (PK id):** agrupa tareas; FK idUsuarioCreador → Usuario.id.

Tarea (PK id): ítem gestionado; FK id Proyecto → Proyecto.id, idUsuarioCreador →

Usuario.id, idDuplicada → Tarea.id (autorreferencia para “Duplicar Tareas”, caso TA-15).

AsignacionTarea (PK compuesta idTarea,idUsuario): unión N:M entre Tarea y Usuario; asignadaPor → Usuario.id.

**Comentario (PK id):** hilo por tarea; FK idTarea → Tarea.id, idUsuarioAutor → Usuario.id (soporta “Comentarios Colaborativos”).

Mencion (PK compuesta idComentario,idUsuarioMencionado): referencia @usuario en comentarios (caso TA-6).

Recurso (PK id): adjuntos/enlaces por tarea (PDF, imagen, video) acorde a la funcionalidad de “recursos educativos”.

Progreso (PK id): % de avance por usuario/tarea (caso “Monitorear Progreso”).

EtiquetaColor / EtiquetaPalabraClave: taxonomías para clasificar tareas (casos TA-12/TA-13).

Puentes TareaEtiquetaColor y TareaEtiquetaPalabraClave (N:M).

**MetaSemanal / MetaMensual:** planificación temporal de objetivos (metas).

**Calificacion (PK id)**: valoración de tareas completadas (caso de uso de calificar)【】.

**Notificacion (PK id):** eventos “mención/cambio/asignación/recordatorio” (caso “Notificar Cambios”).

**RegistroTiempo (PK id):** minutos invertidos por usuario/tarea (caso de registrar tiempo).

**Exportacion (PK id):** exportes a PDF/Excel (casos TA-22/TA-23/TA-24).

**SincronizacionCalendario (PK id):** integración con Google/Outlook (caso de sincronización).

**Anclado (PK compuesta idUsuario,idTarea):** tareas fijadas (caso “Anclar Tareas”).

#### 5.5 Reglas de Integridad Referencial (con acciones ON DELETE)

Proyecto ↔ Usuario (creador): Proyecto.idUsuarioCreador → Usuario.id RESTRICT.

Tarea ↔ Proyecto: Tarea.idProyecto → Proyecto.id CASCADE (al borrar proyecto, se eliminan sus tareas).

Tarea ↔ Usuario (creador): Tarea.idUsuarioCreador → Usuario.id RESTRICT.

Tarea (duplicada): Tarea.idDuplicada → Tarea.id SET NULL.

AsignacionTarea: (idTarea → Tarea.id) CASCADE, (idUsuario → Usuario.id) CASCADE, asignadaPor → Usuario.id RESTRICT.

Comentario: idTarea → Tarea.id CASCADE, idUsuarioAutor → Usuario.id RESTRICT.

Mencion: idComentario → Comentario.id CASCADE, idUsuarioMencionado → Usuario.id CASCADE.

Recurso/Progreso/RegistroTiempo/Calificacion/Notificacion/SincronizacionCalendari o/Anclado: CASCADE en FKs a Tarea; CASCADE en FKs a Usuario cuando corresponda.

Etiquetas y puentes: TareaEtiqueta\* CASCADE hacia Tarea y la tabla de Etiqueta.

Metas: Meta\* .idUsuarioCreador → Usuario.id RESTRICT.

Exportacion: idUsuario → Usuario.id CASCADE.

Estas reglas materializan los flujos y expectativas del sistema

(recuperación/duplicación de tareas, comentarios y menciones, etiquetado, metas, calificación, notificaciones, exportes y sincronizaciones) descritos en los casos de uso.

#### 5.6 Colecciones (NoSQL)

El documento de Etapa 2 contempla el uso de SQL y NoSQL y el dominio incluye funcionalidades de tiempo real, auditoría y colas de trabajo donde NoSQL encaja especialmente bien. A continuación, se define una capa NoSQL complementaria (p.ej., MongoDB) para necesidades no relacionales y de alta frecuencia.

activity\_log (eventos de auditoría/aplicación)

Uso: trazabilidad fina de cambios (quién, cuándo, qué), soporte de “historial” y detección de incidencias.

Ejemplo (JSON):

{

"\_id": "67b1…",

"ts": "2025-10-31T13:40:00Z",

"actorId": "uuid-usuario",

"type": "tarea.update",

"entity": { "kind": "tarea", "id": "uuid-tarea" },

"diff": { "estado": ["en\_progreso","completada"] },

"metadata": { "ip": "x.y.z.w", "ua": "Chrome" }

}

Índices: { ts: -1 }, { actorId: 1, ts: -1 }.

TTL opcional: para logs antiguos (p.ej., 365 días).

notifications\_outbox (patrón outbox para notificaciones)

Uso: garantizar entrega de Notificaciones (email/push/websocket) para eventos como menciones y cambios.

Ejemplo (JSON):

{

"\_id": "67b2…",

"userId": "uuid-usuario",

"taskId": "uuid-tarea",

"kind": "mencion|cambio|asignacion|recordatorio",

"payload": { "comentarioId": "uuid-comentario" },

"status": "pending|sent|failed",

"createdAt": "2025-10-31T13:45:00Z",

"retries": 0

}

Índices: { userId: 1, status: 1, createdAt: -1 }.

calendar\_tokens (credenciales por proveedor)

Uso: almacenar tokens OAuth/refresh cifrados para la sincronización con Google/Outlook.

Ejemplo (JSON):

{

"\_id": "userId::provider", "userId": "uuid-usuario",

"provider": "google|outlook",

"accessTokenEnc": "…", "refreshTokenEnc": "…",

"scope": ["calendar.read","calendar.write"],

"expiresAt": "2025-11-01T10:00:00Z",

"createdAt": "2025-10-31T13:50:00Z"

}

Índices: { userId: 1, provider: 1 } (único).

search\_index (denormalizado para búsqueda rápida)

Uso: indexar texto de tareas, comentarios y etiquetas para filtros y búsqueda (TA-12/13).

Documento (JSON):

{

"\_id": "tarea::uuid",

"kind": "tarea",

"title": "Diseñar rúbrica evaluación",

"body": "…descripcion…",

"labels": ["prioridad:alta","color:#FF9900","palabra:rubrica"],

"project": "uuid-proyecto",

"assignees": ["uuid-1","uuid-2"],

"due": "2025-11-10",

"updatedAt": "2025-10-31T13:55:00Z"

}

Índices: texto compuesto; { updatedAt: -1 }.

jobs\_queue (tareas asíncronas: exportes, importaciones, sincronizaciones)

Uso: soportar Exportar (PDF/Excel) y Sincronizar Calendario sin bloquear la UI. Ejemplo (JSON):

{

"\_id": "job-uuid",

"type": "export.pdf|export.excel|sync.calendar",

"args": { "userId": "uuid", "filters": {...} },

"status": "queued|running|done|failed",

"progress": 35,

"createdAt": "2025-10-31T14:00:00Z",

"updatedAt": "2025-10-31T14:02:00Z",

"error": null

}

Índices: { status: 1, createdAt: -1 }.

presence\_sessions (tiempo real)

Uso: presencia/sesiones WebSocket para indicar usuarios activos en tareas/comentarios.

TTL: 10–30 minutos; índices por userId y taskId.

### Documentación del Backend: Task Assistant (En Progreso)

1. Resumen y Arquitectura

La API está siendo desarrollada con una arquitectura modular utilizando NestJS, diseñada para ser robusta, escalable y segura.

La lógica de negocio está separada en Módulos, donde cada módulo encapsula un recurso (ej. Tareas, Comentarios). La interacción con la base de datos PostgreSQL se gestiona a través del ORM Prisma, y la seguridad de los endpoints se maneja mediante JSON Web Tokens (JWT) y un sistema de Control de Acceso Basado en Roles (RBAC).

1. Stack Tecnológico

* Framework: NestJS (Node.js)

* Lenguaje: TypeScript

* Base de Datos: PostgreSQL

* ORM: Prisma

* Autenticación: JWT (usando @nestjs/jwt y passport-jwt)

* Validación: class-validator y DTOs (Data Transfer Objects)

3. Módulo de Autenticación (/auth)

Este módulo gestiona el inicio de sesión y la estrategia de seguridad de la API.

Componentes Clave

* auth.controller.ts: Expone el endpoint público de login.

* auth.service.ts:

* 1. validateUser(): Compara el email y la contraseña (usando bcrypt) contra la tabla Usuario.

○ login(): Si la validación es exitosa, genera y firma un access\_token (JWT).

○

* jwt.strategy.ts:

* 1. Valida los JWT entrantes en cada petición protegida.

○ Extrae el payload del token (que contiene userId y rol) y lo adjunta al objeto request.user, haciéndolo disponible en todos los controladores.

* jwt-auth.guard.ts: Guardia global que protege los endpoints. Si una petición no tiene un JWT válido, la rechaza (Error 401 Unauthorized).

Endpoints Implementados

* POST /auth/login

* 1. Acción: Autentica a un usuario.

○ Body: { "email": "...", "password": "..." }

○ Respuesta: { "access\_token": "..." }

4. Control de Acceso Basado en Roles (RBAC)

Implementamos un sistema de RBAC personalizado para cumplir con los requisitos del documento (ej. "solo Administrador y Docente Principal").

* @Roles() (Decorador): Un decorador (src/auth/decorators/roles.decorator.ts) que se aplica a los controladores para especificar qué roles tienen permiso.

○ Ejemplo: @Roles(RolUsuario.administrador,

RolUsuario.docente\_principal)

* RolesGuard (Guardia): Un guardia (src/auth/guards/roles.guard.ts) que se activa después del JwtAuthGuard. Compara el rol del usuario (obtenido del request.user) con los roles definidos en el decorador @Roles(). Si no coinciden, rechaza la petición (Error 403 Forbidden).

5. Módulo de Tareas (/proyectos/:proyectoId/tareas)

Este es el módulo central de la aplicación. Gestiona el ciclo de vida completo de las tareas, anidado dentro de un proyecto.

DTOs (Data Transfer Objects)

* create-tarea.dto.ts: Valida los datos para crear una tarea (título, descripción, etc.).

* update-tarea.dto.ts: Valida los datos para actualizar una tarea (todos opcionales).

* assign-tarea.dto.ts: Valida el idUsuario (del estudiante) al asignar una tarea.

* update-progreso.dto.ts: Valida el porcentaje (0-100) para el progreso.

* log-tiempo.dto.ts: Valida los minutos a registrar.

Endpoints Implementados (en tareas.controller.ts)

* POST /proyectos/:proyectoId/tareas

* 1. Acción: (RF-001) Crear una nueva tarea.

○ Guards: JwtAuthGuard, RolesGuard (Admin, DocentePrincipal).

* GET /proyectos/:proyectoId/tareas

* 1. Acción: (RF-001) Listar todas las tareas activas de un proyecto.

○ Guards: JwtAuthGuard.

* GET /proyectos/:proyectoId/tareas/:tareaId

* 1. Acción: (RF-001) Obtener detalles de una tarea.

○ Guards: JwtAuthGuard.

* PATCH /proyectos/:proyectoId/tareas/:tareaId

* 1. Acción: (RF-005) Actualizar una tarea.

○ Guards: JwtAuthGuard, RolesGuard (Admin, DocentePrincipal).

* DELETE /proyectos/:proyectoId/tareas/:tareaId

* 1. Acción: (RF-001) Borrado lógico (Soft Delete) de una tarea.

○ Guards: JwtAuthGuard, RolesGuard (Admin, DocentePrincipal).

* POST /proyectos/:proyectoId/tareas/:tareaId/recuperar

* 1. Acción: (RF-002) Restaurar una tarea desde la papelera.

○ Guards: JwtAuthGuard, RolesGuard (Admin, DocentePrincipal).

* POST /proyectos/:proyectoId/tareas/:tareaId/asignar

* 1. Acción: (RF-006) Asignar una tarea a un usuario.

○ Guards: JwtAuthGuard, RolesGuard (Admin, DocentePrincipal).

* PATCH /proyectos/:proyectoId/tareas/:tareaId/progreso

* 1. Acción: (RF-011) Actualizar el porcentaje de progreso.

○ Guards: JwtAuthGuard.

* POST /proyectos/:proyectoId/tareas/:tareaId/tiempo

* 1. Acción: (RF-012) Registrar tiempo en una tarea.

○ Guards: JwtAuthGuard.

Lógica de Servicio (en tareas.service.ts)

* remove(): Implementa un Soft Delete actualizando el campo eliminada de la tarea con la fecha actual. No borra el registro.

* recover(): Revierte el Soft Delete poniendo el campo eliminada en null.

* assign(): Crea una nueva entrada en la tabla de unión AsignacionTarea.

* updateProgreso(): Es una operación compleja:

* 1. Verifica que el usuario esté asignado a la tarea (checkIfUserIsAssigned).

* 1. Usa prisma.progreso.upsert() para crear o actualizar el registro de progreso individual del usuario.

* 1. Actualiza el estado de la Tarea principal (ej. a completada si el porcentaje es 100).

* 1. Envuelve ambas operaciones en una transacción ($transaction) para garantizar la integridad de los datos.

* logTiempo():

* 1. Verifica que el usuario esté asignado (checkIfUserIsAssigned).

* 1. Crea una nueva entrada en la tabla RegistroTiempo.

6. Módulo de Comentarios (.../:tareaId/comentarios)

Gestiona la colaboración y las menciones dentro de una tarea.

DTOs

* create-comentario.dto.ts: Valida el contenido (texto) y un array opcional menciones (lista de IDs de usuarios).

Endpoints Implementados (en comentarios.controller.ts)

* POST /proyectos/:proyectoId/tareas/:tareaId/comentarios

○ Acción: (RF-008, RF-009) Publicar un nuevo comentario y procesar menciones.

○ Guards: JwtAuthGuard.

Lógica de Servicio (en comentarios.service.ts)

* create():

* 1. Debe verificar que el usuario tenga acceso a la tarea.

* 1. Usa prisma.$transaction para asegurar que el comentario y todas sus menciones se creen de forma atómica.

* 1. Paso 1 (Transacción): Crea la entrada en la tabla Comentario.

* 1. Paso 2 (Transacción): Si el DTO incluye un array menciones, crea las entradas correspondientes en la tabla Mencion.

* 1. Aquí se debe disparar la notificación RF-021.

**ETAPA 2: PERSISTENCIA DE DATOS CON BACKEND**

### INTRODUCCIÓN

Esta etapa del proyecto "Task Assistant" se centra en la construcción de la "sala de máquinas" de la aplicación. Es la fase donde se implementa la lógica del servidor (backend) y la conexión con la base de datos (persistencia).

### PROPÓSITO DE LA ETAPA

El propósito principal es implementar la capa de servicios que permite a la aplicación almacenar, gestionar y exponer de forma segura toda la lógica de negocio de Task Assistant. Esto incluye crear la API REST, gestionar la autenticación de usuarios y conectarse a un sistema de base de datos para realizar todas las operaciones CRUD (Crear, Leer, Actualizar, Borrar) sobre las tareas, usuarios, comentarios y otros recursos.

### ALCANCE DE LA ETAPA

El alcance de esta etapa cubre:

* La implementación de la base de datos.
* El desarrollo de una API RESTful para gestionar todos los casos de uso definidos (CRUD de tareas, asignación, comentarios, calificación, etc.).
* La implementación de un sistema de autenticación y autorización basado en tokens (JWT) y roles (RBAC).
* La creación de la lógica para el manejo de recursos, notificaciones y sincronización básica.

### DEFINICIONES Y ACRÓNIMOS

* API (Application Programming Interface): Interfaz que permite la comunicación entre el frontend y el backend.
* Backend: Lado del servidor de la aplicación, responsable de la lógica de negocio y la persistencia de datos.
* CRUD: Operaciones básicas de gestión de datos: Crear, Leer, Actualizar, Borrar.
* JWT (JSON Web Token): Estándar para crear tokens de acceso que validan la identidad de un usuario.
* NestJS: Framework de Node.js progresivo para construir aplicaciones del lado del servidor eficientes y escalables.
* NoSQL: Base de datos no relacional (como MongoDB), que almacena datos en documentos flexibles tipo JSON.

* ORM (Object-Relational Mapping): Herramienta (Prisma) que mapea los

modelos de la base de datos a objetos en el código.

* Prisma: ORM moderno para Node.js y TypeScript.
* RBAC (Role-Based Access Control): Sistema de seguridad que asigna permisos basado en el rol del usuario (ej. Administrador, Estudiante).
* REST (Representational State Transfer): Arquitectura para diseñar APIs web.

### DISEÑO DE LA ARQUITECTURA DE BACKEND DESCRIPCIÓN DE LA ARQUITECTURA PROPUESTA

La arquitectura elegida es una Arquitectura Modular basada en Dominios utilizando el framework NestJS. Esta arquitectura es robusta y escalable, alineada con las mejores prácticas de desarrollo de software.

El backend se organiza en módulos que encapsulan una responsabilidad de negocio específica (ej. Auth, Tareas, Comentarios, Metas). Cada módulo contiene su propio:

**Controlador (controller):** Define los endpoints de la API (las URLs) y maneja las peticiones (request) y respuestas (response).

**Servicio (service):** Contiene la lógica de negocio pura (ej. "al marcar 100% de progreso, cambiar el estado a 'completada'").

**Módulo (module):** Conecta el controlador, el servicio y cualquier dependencia (como el PrismaModule).

### COMPONENTES DEL BACKEND

Los principales módulos implementados o diseñados son:

**AuthModule:** Responsable del login (POST /auth/login), validación de usuarios y generación de tokens JWT.

**TareasModule:** Módulo central que gestiona el CRUD de tareas y todas las interacciones relacionadas (asignar, progreso, tiempo, calificar, etc.).

**ComentariosModule:** Gestiona la creación de comentarios y menciones en las tareas.

**MetasModule**: Gestiona la creación y vinculación de metas semanales/mensuales a las tareas.

**EtiquetasModule:** Gestiona la lista maestra de etiquetas disponibles (color y palabra clave).

**PrismaModule:** Módulo reutilizable que provee la conexión a la base de datos a todos los demás servicios.

### DIAGRAMAS DE ARQUITECTURA

La arquitectura de la API sigue un flujo estándar de NestJS:

Un cliente (Frontend) envía una petición HTTP (ej. POST /tareas). El JwtAuthGuard intercepta la petición para verificar el token JWT.

El RolesGuard (si aplica) verifica el rol (admin, docente) dentro del token.

Si la seguridad pasa, la petición llega al Controlador (TareasController).

El Controlador llama al Servicio (TareasService).

El Servicio ejecuta la lógica de negocio y usa Prisma Client para consultar la base de datos MongoDB.

Prisma devuelve los datos al Servicio, el Servicio al Controlador, y el Controlador al Cliente.

### ELECCIÓN DE LA BASE DE DATOS EVALUACIÓN DE OPCIONES (SQL O NOSQL)

**SQL (PostgreSQL):** La especificación original del PDF recomendaba PostgreSQL. Las bases de datos SQL son excelentes para datos altamente estructurados, transacciones complejas y consistencia de datos (ACID).

**NoSQL (MongoDB):** Las bases de datos NoSQL son flexibles, escalables horizontalmente y excelentes para manejar datos semi-estructurados o que evolucionan rápidamente. Su modelo de "documentos" (similar a JSON) se alinea muy bien con el desarrollo en JavaScript/TypeScript.

### JUSTIFICACIÓN DE LA ELECCIÓN

Aunque la documentación inicial sugería PostgreSQL, la implementación actual utiliza MongoDB (NoSQL), basado en el schema.prisma proporcionado durante el desarrollo.

Esta elección se justifica por:

Flexibilidad de Esquema: Permite que las tareas y otros documentos evolucionen sin migraciones complejas.

Ecosistema JavaScript: Al usar NestJS (Node.js), trabajar con documentos BSON/JSON de MongoDB es nativo y reduce la fricción de desarrollo.

Refactorización: Como se demostró, migrar de un modelo de "muchos a muchos" (SQL) a un modelo de "documento anidado" o "referencia 1:N" (Mongo) fue rápido (ej. AsignacionTarea vs. asignadoId).

### DISEÑO DE ESQUEMA DE BASE DE DATOS

El esquema de la base de datos está definido en el archivo schema.prisma y utiliza el conector de MongoDB. Los modelos centrales definidos incluyen:

**Usuario:** Almacena email, contraseña (hash), nombre y rol.

**Tarea:** El modelo central. Almacena título, estado, prioridad, progreso, tiempoRegistrado y mantiene referencias directas (creadorId, asignadoId).

**Comentario:** Vinculado a una Tarea y un Usuario (autor).

**EtiquetaPalabra, EtiquetaColor, EtiquetaTarea:** Implementan el sistema de etiquetado (N:M).

**Meta, MetaTarea:** Implementan el sistema de metas (N:M).

**Calificacion:** Vinculada a una Tarea y un Usuario (calificador). **Recurso:** Vinculado a una Tarea (para enlaces y archivos).

### IMPLEMENTACIÓN DEL BACKEND ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN

El lenguaje elegido es TypeScript. Se utiliza en conjunto con el framework NestJS. Esta elección proporciona tipado estático (menos errores), características modernas de ES6+ y un ecosistema robusto (NPM).

### CREACIÓN DE LA LÓGICA DE NEGOCIO

La lógica de negocio reside en los archivos \*.service.ts (ej. tareas.service.ts). Ejemplos de lógica implementada:

**RF-011 (Progreso):** Al actualizar el progreso, el servicio cambia automáticamente el estado de la tarea (ej. 100% = 'completada').

**RF-012 (Tiempo):** El servicio utiliza un increment atómico de Prisma para sumar el tiempo registrado.

**RF-020 (Calificación):** El servicio primero valida que la tarea.estado sea 'completada' antes de permitir la creación de una Calificacion.

**Autorización:** Se implementó un helper checkIfUserIsAssigned para asegurar que un estudiante solo pueda registrar tiempo o progreso en tareas que le pertenecen.

### DESARROLLO DE ENDPOINTS Y APIS

Se han desarrollado controladores modulares (TareasController, MetasController, etc.) para exponer la API REST. Los endpoints siguen las convenciones REST, usando verbos HTTP (GET, POST, PATCH, DELETE) y rutas anidadas para expresar relaciones (ej. POST /tareas/:tareaId/calificar).

### AUTENTICACIÓN Y AUTORIZACIÓN

Se ha implementado un robusto sistema de seguridad:

**Autenticación (JWT):** Se usa AuthModule con passport-jwt. El endpoint POST /auth/login valida al usuario y firma un token JWT. Todas las demás rutas están protegidas por el JwtAuthGuard.

**Autorización (RBAC):** Se crearon RolesGuard y el decorador @Roles(). Los endpoints que requieren permisos específicos (ej. crear tareas) se decoran con @Roles('administrador', 'docente\_principal'), bloqueando el acceso a otros roles como 'estudiante'.

### CONEXIÓN A LA BASE DE DATOS CONFIGURACIÓN DE LA CONEXIÓN

La conexión se gestiona mediante Prisma. El archivo schema.prisma define el datasource para mongodb y la variable de entorno DATABASE\_URL. El PrismaService inyecta el PrismaClient en los servicios que lo necesitan.

### DESARROLLO DE OPERACIONES CRUD

Todas las operaciones CRUD se manejan a través de los métodos de Prisma Client:

prisma.tarea.create(...)

prisma.tarea.findMany(...)

prisma.tarea.findUnique(...)

prisma.tarea.update(...)

prisma.tarea.delete(...)

prisma.tarea.update({ data: { tiempoRegistrado: { increment: ... } } }) (Operación atómica)

### MANEJO DE TRANSACCIONES

Para operaciones que requieren múltiples escrituras (como crear un comentario y sus menciones), se utiliza prisma.$transaction([...]). Esto asegura que si una parte de la operación falla, todas las partes se revierten, manteniendo la integridad de los datos.

**PRUEBAS DEL BACKEND**

### DISEÑO DE CASOS DE PRUEBA

Los casos de prueba se diseñan en base a los Requisitos Funcionales (RF) y los roles:

**Prueba de Éxito:** Un Docente puede crear una tarea (POST /tareas). **Prueba de Fracaso (Autorización):** Un Estudiante no puede crear una tarea (POST /tareas debe devolver 403 Forbidden).

**Prueba de Lógica:** Si un Estudiante pone el progreso al 100% (PATCH

/tareas/:id/progreso), verificar que el estado de la tarea cambie a 'completada'. **Prueba de Error (Validación):** Si se envía un email inválido al crear un usuario, debe devolver 400 Bad Request.

### EJECUCIÓN DE PRUEBAS UNITARIAS Y DE INTEGRACIÓN

**Pruebas Unitarias:** (Pendiente) Se deben enfocar en los \*.service.ts para probar la lógica de negocio de forma aislada (mockeando Prisma).

**Pruebas de Integración (E2E):** (Pendiente) Se deben probar los \*.controller.ts para simular peticiones HTTP reales y verificar todo el flujo (Guardia -> Controlador -> Servicio -> Base de Datos) usando el entorno de pruebas de NestJS.

### MANEJO DE ERRORES Y EXCEPCIONES

El backend utiliza el sistema de excepciones de NestJS para manejar errores:

**404 NotFoundException:** Se lanza si un usuario intenta buscar un recurso que no existe (ej. prisma.tarea.findUnique falla).

**403 ForbiddenException:** Se lanza si RolesGuard falla, o si un usuario intenta realizar una acción no permitida (ej. calificar una tarea no completada).

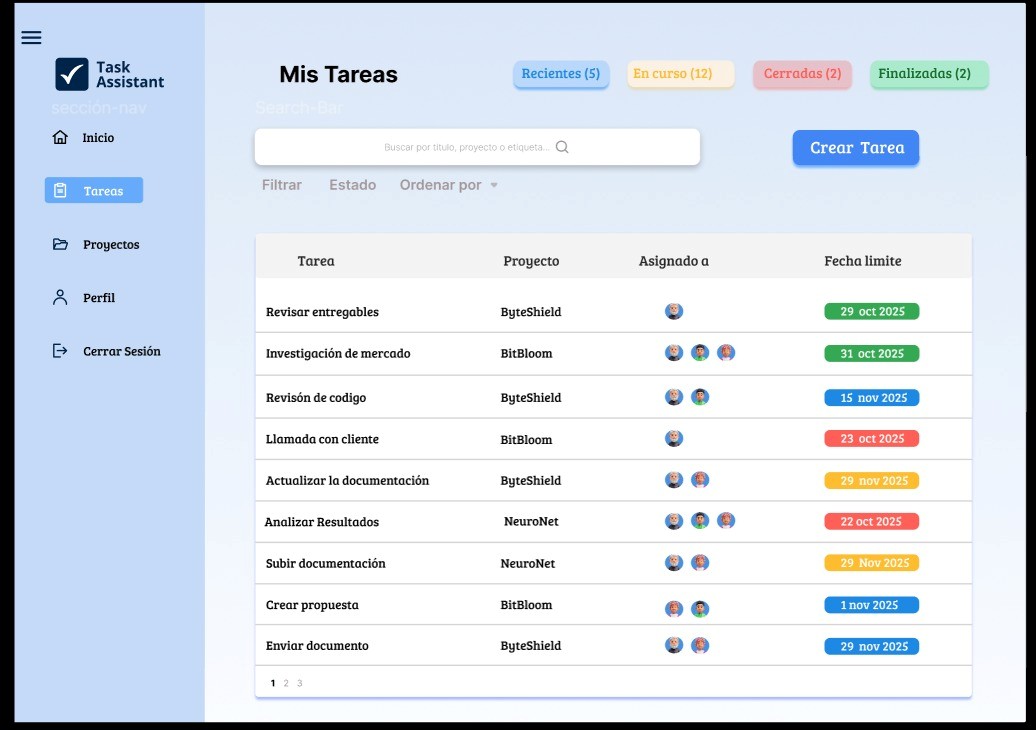
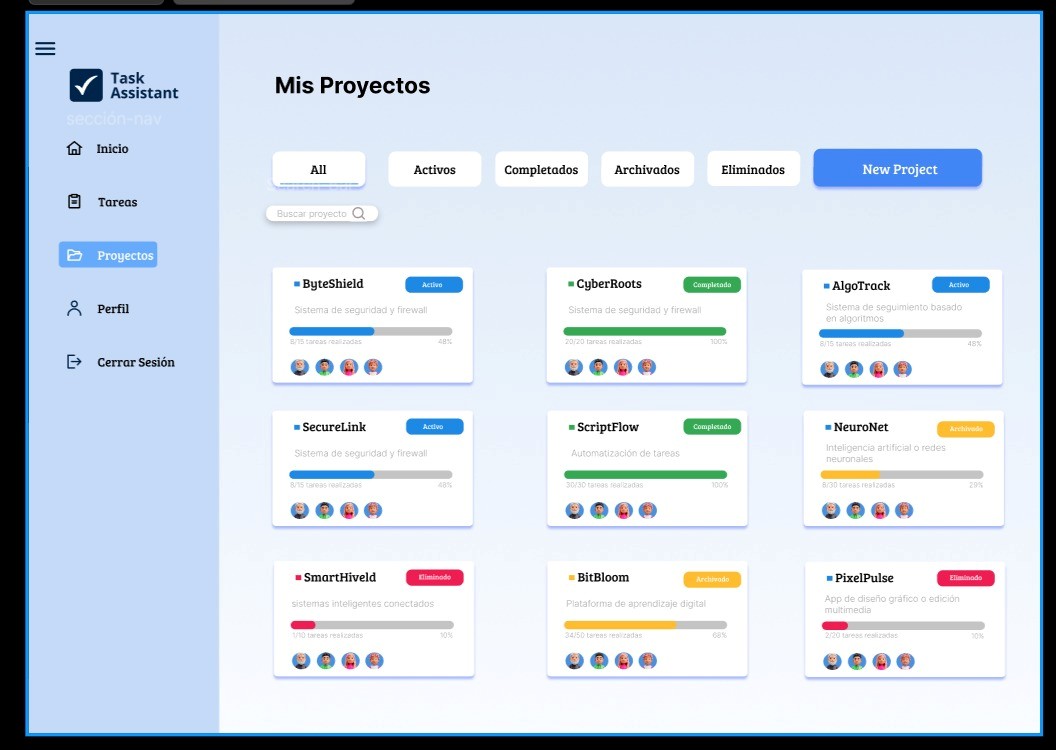
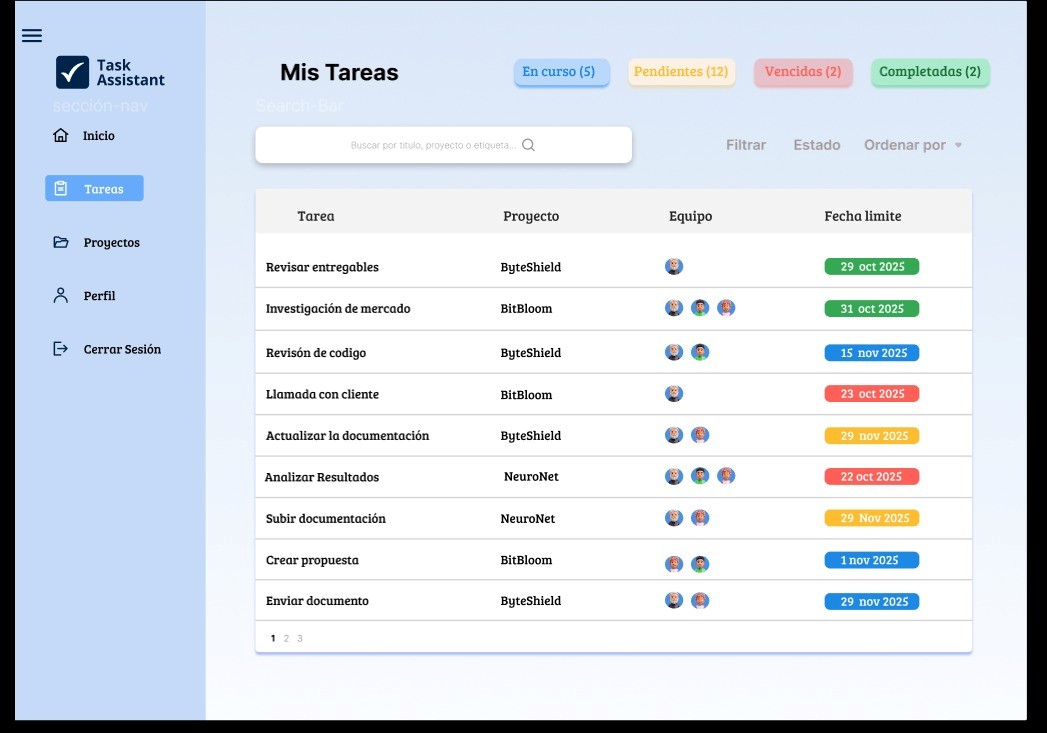
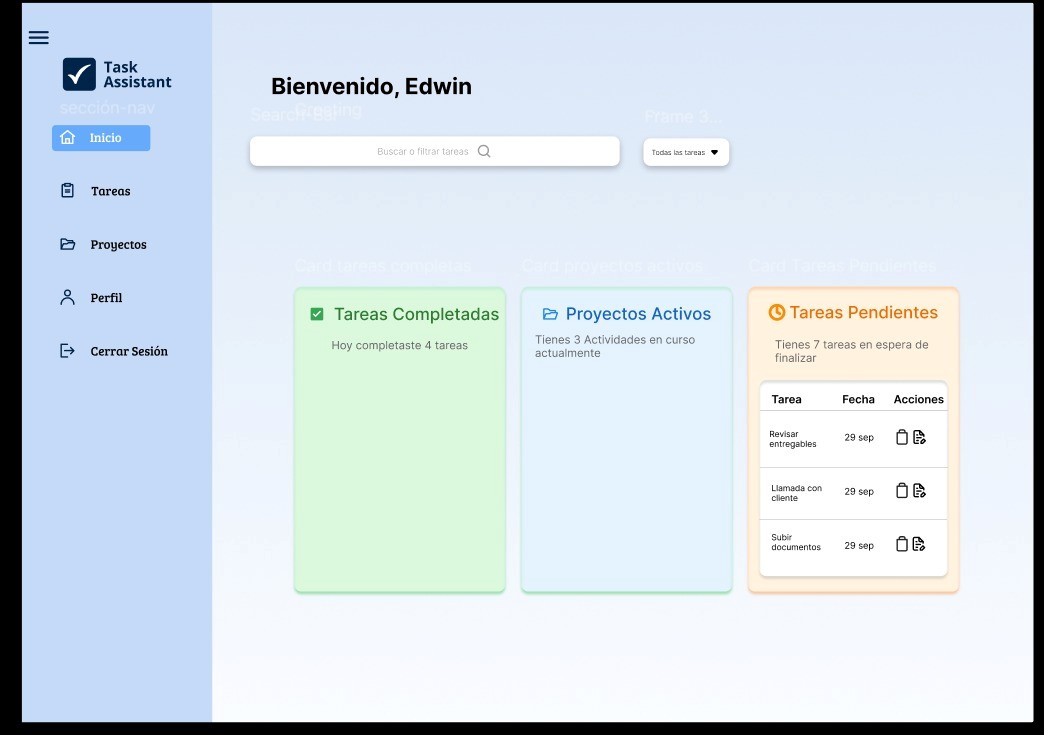
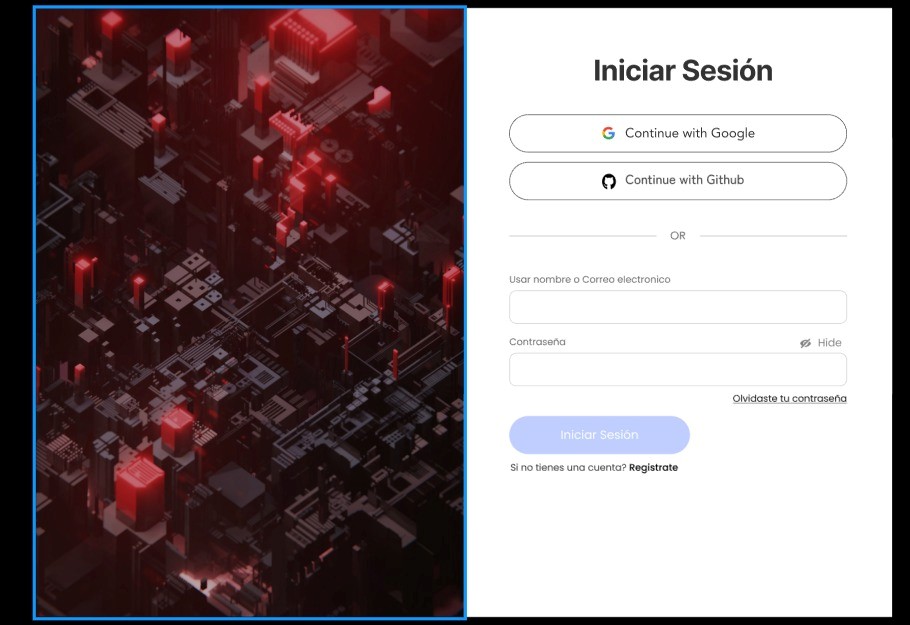
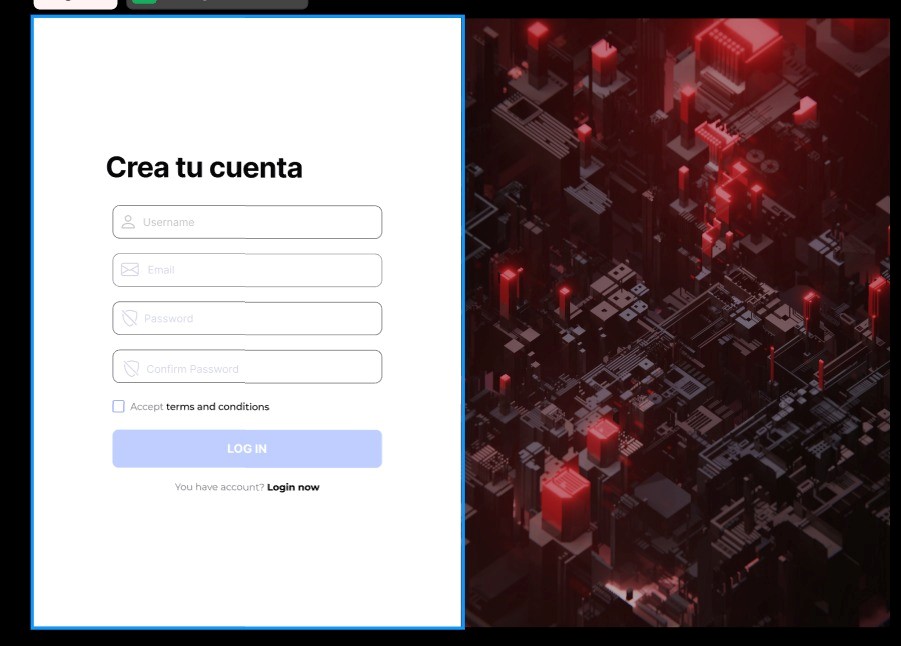
**401 UnauthorizedException**: Lanzada automáticamente por JwtAuthGuard si el token falta o es inválido.

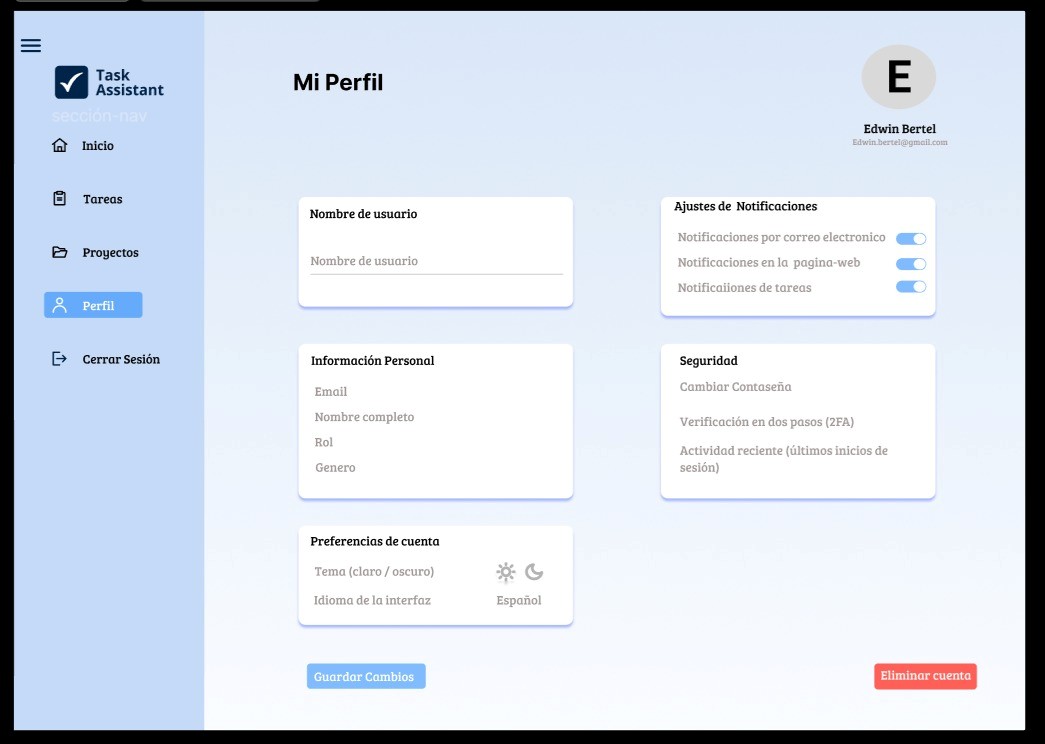
**400 BadRequestException:** Lanzada automáticamente por los DTOs si la validación de class-validator falla (ej. un email no es un email).

**Manejo de Errores de Prisma:** Se capturan códigos de error específicos de Prisma (ej. P2002 por duplicados, P2025 por registros no encontrados) para devolver mensajes de error claros.

## Anexos







### ETAPA 3: CONSTRUCCIÓN DEL CLIENTE (FRONTEND) INTRODUCCIÓN

En esta etapa se desarrolló la interfaz de usuario (UI) y la experiencia de usuario (UX) de la plataforma Task Assistant. El cliente web consume la API RESTful desarrollada en la etapa anterior, permitiendo a los usuarios interactuar con el sistema de manera intuitiva y segura.

### PROPÓSITO DE LA ETAPA

El objetivo fue construir una Aplicación Web Progresiva (PWA) responsiva que materialice los requisitos funcionales. Se buscó desacoplar totalmente el frontend del backend, permitiendo un desarrollo independiente y escalable.

### ALCANCE DE LA ETAPA

El desarrollo abarcó :

Configuración del entorno de desarrollo con Nuxt 3.

Implementación de diseño visual utilizando Vuetify 3 (Material Design).

Gestión de estado global (sesión de usuario) con Pinia.

Consumo de servicios del backend (API) mediante useFetch.

Implementación de vistas para todos los módulos: Autenticación, Tareas, Metas, Usuarios y Etiquetas.

Protección de rutas mediante Middleware.

### ARQUITECTURA DEL FRONTEND STACK TECNOLÓGICO

**Framework Principal:** Nuxt 3 (basado en Vue 3). Se eligió por su renderizado híbrido, enrutamiento automático y optimización de rendimiento.

**Biblioteca de UI:** Vuetify 3. Proporciona componentes pre-construidos (tarjetas, tablas, formularios) basados en Material Design, acelerando el desarrollo y garantizando consistencia visual.

**Gestión de Estado:** Pinia. Utilizado para almacenar datos globales como el token JWT y el perfil del usuario logueado.

**Cliente HTTP:** $fetch / useFetch (Nativo de Nuxt). Para la comunicación asíncrona con el backend NestJS.

**Lenguaje:** JavaScript / TypeScript (Composition API).

### ESTRUCTURA DEL PROYECTO

La aplicación sigue la estructura de directorios estándar de Nuxt:

**/pages:** Define las rutas de la aplicación (ej. /login, /dashboard, /tareas/[id]).

**/layouts:** Define las plantillas maestras (default.vue para la app con barra lateral, blank.vue para login).

**/stores:** Contiene la lógica de estado global (auth.ts).

**/middleware:** Contiene la lógica de protección de rutas (auth.ts).

**nuxt.config.ts:** Archivo de configuración central (módulos, variables de entorno).

### IMPLEMENTACIÓN DE MÓDULOS 1. MÓDULO DE AUTENTICACIÓN

**Vistas:** Login y Registro.

**Lógica:**

El usuario ingresa credenciales.

El frontend envía POST /auth/login al backend.

Si es exitoso, el backend devuelve un JWT.

Pinia Store (auth.ts) guarda este token en localStorage para persistencia.

Middleware (middleware/auth.ts) verifica la existencia del token antes de permitir el acceso a rutas protegidas.

### 2. DASHBOARD Y NAVEGACIÓN

**Layout (layouts/default.vue):** Implementa una barra de navegación lateral (v-navigation-drawer) que se adapta según el rol del usuario (Admin, Docente, Estudiante).

**Dashboard (pages/dashboard.vue):** Muestra tarjetas resumen con estadísticas rápidas y accesos directos.

### 3. GESTIÓN DE TAREAS

**Listado (pages/tareas/index.vue):** Visualización de tareas en cuadrícula de tarjetas (v-card). Incluye filtros visuales por estado.

**Creación (pages/tareas/crear.vue):** Formulario validado para ingresar título, descripción y fecha. Solo accesible para roles autorizados.

**Detalle (pages/tareas/[id].vue):** Vista completa que permite:

Ver detalles.

Actualizar progreso (Slider interactivo).

Registrar tiempo.

Asignar estudiantes (Selector dinámico).

Comentar en tiempo real.

### 4. GESTIÓN DE METAS Y PLANIFICACIÓN

**Vistas (pages/metas/index.vue):** Uso de pestañas (v-tabs) para separar metas Semanales de Mensuales.

**Funcionalidad:** Permite crear metas y vincular tareas existentes a ellas mediante un diálogo modal.

### 5. ADMINISTRACIÓN (USUARIOS Y ETIQUETAS)

**Usuarios (pages/usuarios/index.vue):** Tabla de datos (v-data-table) para que el Administrador gestione el alta de Docentes y Estudiantes.

**Etiquetas (pages/etiquetas/index.vue):** Interfaz para gestionar la "lista maestra" de etiquetas de colores y palabras clave.

### INTEGRACIÓN Y SEGURIDAD

Variables de Entorno: La URL de la API no está "hardcodeada", se configura mediante NUXT\_PUBLIC\_API\_BASE en nuxt.config.ts, permitiendo cambiar fácilmente entre desarrollo (localhost:3000) y producción.

**Manejo de Errores:** Se implementaron bloques try/catch en todas las llamadas a la API para mostrar alertas visuales (v-alert) al usuario en caso de fallos de conexión o errores de validación (400/403/500).

**Control de Acceso en Cliente:** Aunque el backend es la autoridad final, el frontend oculta botones y menús (usando v-if y computed properties) según el rol del usuario para mejorar la experiencia (ej. un estudiante no ve el botón "Borrar Tarea").

### CONCLUSIÓN DEL DESARROLLO

Se ha logrado una aplicación funcional, visualmente atractiva y arquitectónicamente separada. El Frontend actúa como una interfaz rica que consume los servicios seguros del Backend, cumpliendo con los requisitos de usabilidad y funcionalidad del proyecto Task Assistant.