Proyecto SIGETAP

📝 Requerimiento del Proyecto

Nombre del Sistema: *SIGETAP* — Sistema Integral de Gestión de Tareas y Proyectos

Stack Tecnológico:

- **Backend**: Python 3.12+, Django 5, Django REST Framework
- **Frontend**: ReactJS 18+
- **Base de datos**: PostgreSQL 14+
- **Autenticación**: Login institucional mediante **LDAP + JWT**

---

1. 🎯 Objetivo General

Desarrollar una aplicación web para gestionar proyectos y tareas asignadas al personal de distintas gerencias y unidades, con control de acceso basado en roles, estructura jerárquica organizacional, y herramientas de seguimiento visual como diagramas de Gantt y reportes de avance. Autenticación segura vía LDAP con auditoría de accesos.

---

2. 📁 Módulos Funcionales

2.1. Gestión de Usuarios y Roles (`core.users`)

- Inicio de sesión vía LDAP con creación automática del usuario al primer acceso.
- Campos organizacionales en perfil:
  - `nom_gerencia_general` , `nom_gerencia` , `nom_unidad` , `nom_unidad_reporta` , `nom_coordinacion` , `nom_departamento`
- Roles del sistema:
  - Administrador General

    - Gerente
    - Coordinador
    - **Líder**
    - Miembro de equipo
    - Observador
- Funcionalidades:
    - Consulta de perfil
    - Asociación a proyectos y tareas

2.2. Estructura Organizativa

- Modelada mediante los campos del perfil del usuario.
- Cada nivel reporta jerárquicamente al siguiente.
- Utilizado para aplicar filtros de visualización y control de acceso.

2.3. Gestión de Proyectos (`core.proyectos`)

- Campos:
    - Nombre, código, unidad responsable, fechas, objetivo, estado
    - `lider`: ForeignKey a usuario líder del proyecto
- Estados: Planificado, En ejecución, Pausado, Finalizado
- Funcionalidades:
    - Creación y edición de proyectos
    - Asignación de participantes y líder
    - Adjuntos (documentos, cronogramas)
    - **Gestión de diagrama de Gantt**
        - Se podrá:
            - Subir un archivo de Gantt (formato PDF o imagen)
            - O bien construir un cronograma visual tipo Gantt (React Gantt Chart)
        - Opcionalmente: definir fases, hitos y relaciones entre tareas

2.4. Gestión de Tareas (`core.tareas`)

- Asociadas a un proyecto y a un usuario responsable
- Campos:
  - Título, descripción, prioridad, fechas, estado, comentarios
- Estados: Por hacer, En progreso, En revisión, Completada, Rechazada
- Funcionalidades:
  - Checklists o subtareas
  - Archivos adjuntos
  - Visibilidad limitada según asignación y rol

2.5. Notificaciones (`core.notificaciones`)

- Envío por correo y notificaciones internas:
  - Nuevas asignaciones
  - Cambios de estado
  - Recordatorios de vencimiento

2.6. Auditoría de Accesos (`core.security`)

- Registro de intentos de acceso (exitosos y fallidos)
- Logs con IP, fecha y estado
- No maneja permisos; solo auditoría

2.7. Autenticación (`core.login`)

- Login exclusivo vía LDAP
- Emisión y refresco de JWT
- Endpoints:
  - `/api/login/` , `/api/token/refresh/` , `/api/logout/`

2.8. API Pública Interna (`core.api`)

- API RESTful con DRF
- Permisos y filtros por rol y jerarquía
- Documentación Swagger/OpenAPI

2.9. Rol de Líder de Proyecto

- El Líder es un usuario asignado por el Gerente o Coordinador dentro de un proyecto.
- Puede:
  - Ver todo el proyecto
  - Asignar tareas a miembros
  - Cambiar estados
  - Subir entregables
  - Acceder a dashboards y reportes
- No puede crear nuevos proyectos ni ver otros ajenos.

2.10. Módulo de Reportes de Avance (`core.reportes`)

- Para cada proyecto:
  - Generar reporte general del avance:
    - % de tareas completadas
    - Tareas en progreso / vencidas
    - Tiempo estimado vs tiempo real
    - Participación de cada usuario
    - Fechas importantes y desviaciones
  - Exportar el resumen como:
    - PDF (con `WeasyPrint` o `xhtml2pdf`)
    - Excel (con `openpyxl`)
  - Accesible para:
    - Líder del proyecto
    - Coordinador o Gerente asignado

- Administrador General

---

3. ⚙️ Requerimientos Técnicos

Backend (Django 5 + DRF)

- Apps modulares en `core/`: `users`, `proyectos`, `tareas`, `notificaciones`, `security`, `login`, `api`
- LDAP + JWT para autenticación
- Serializers, filtros y permisos bien definidos
- Auditoría de accesos centralizada

Frontend (ReactJS)

- Login institucional (LDAP) con JWT
- Interfaces:
  - Gestión de proyectos/tareas
  - Visualización Kanban
  - Visualización Gantt (ej: `frappe-gantt`, `react-gantt`)
  - Dashboards y reportes
- Librerías sugeridas:
  - Axios, React Hook Form, Recharts, SweetAlert2, Chart.js

---

4. 📈 Control de Acceso a Proyectos y Tareas

Rol	Acceso permitido
Miembro de equipo	Solo proyectos y tareas a los que está asignado
Coordinador	Todos los proyectos y tareas dentro de su coordinación
Gerente	Todos los proyectos y tareas dentro de su gerencia y subordinados
Líder	Acceso completo a los proyectos que lidera (gestión de tareas, Gantt, reportes)

Administrador General	Acceso completo a todo

5. 🗂️ Estructura del backend

```
backend/
├── core/
│   ├── __init__.py
│   ├── users/          # Gestión de usuarios y jerarquía organizacional
│   ├── proyectos/      # Proyectos, Gantt, liderazgo
│   ├── tareas/         # Tareas, subtareas, comentarios
│   ├── reportes/       # Generación de reportes de avance, PDF, Excel
│   ├── notificaciones/  # Alertas internas y por correo
│   ├── security/       # Auditoría de accesos (login)
│   ├── login/          # Autenticación LDAP + JWT
│   └── api/            # Rutas DRF y Swagger
├── config/             # settings.py, urls.py, asgi.py, wsgi.py
└── manage.py
```