1. Objetivo general

Diseñar una aplicación que permita planificar, organizar y hacer seguimiento de varios proyectos, desglosados en trabajos y actividades, con control de estados, fechas, responsables y progreso del proyecto, trabajos y actividad.

La estructura sería jerárquica:
Proyecto → Trabajo → Actividad

2. Entidades principales

Modelo: Proyecto

Representa un conjunto de trabajos.

Campos propuestos:

Nombre del proyecto, Descripción general del proyecto, Fecha de inicio, Fecha de fin, estado del proyecto, responsable del proyecto, porcentaje de avance respecto a partir de trabajos y actividades, relación con trabajos del proyecto, Avance individual (0–100%).

Ayuda: Puedes usar un widgets de progreso de Odoo que harán tu aplicación más visual.

Posibles estados para el proyecto:

- Borrador
- En planificación
- En ejecución
- Finalizado
- Cancelado

Modelo: Trabajo

Cada trabajo pertenece a un proyecto y agrupa actividades relacionadas.

Campos propuestos: Descripción del trabajo, Proyecto al que pertenece, Responsable o técnico asignado, Fecha de inicio, Fecha de finalización, Estado actual del trabajo (ver estados), Promedio de avance de las actividades, importancia de las actividades (Baja / Media / Alta / Urgente), Avance individual (0–100%)

Posibles estados:

- Pendiente

- En progreso
- En revisión
- Finalizado

Modelo: Actividad

Las unidades más pequeñas de trabajo que forman parte de un trabajo.

Campos propuestos: Nombre de la actividad, Detalle de la tarea, Trabajo al que pertenece, Persona que la realiza, Inicio planificado, Fin planificado, Fin planificado, Estado de la actividad, Avance individual (0–100%)

Posibles estados:

- Pendiente
- En curso
- En revisión
- Finalizada
- Cancelada

3. Reglas de negocio y comportamiento

1. **Avance automático:**
   - El avance del *trabajo* = media ponderada de las actividades.
   - El avance del *proyecto* = media ponderada de los trabajos.
2. **Estados automáticos:**
   - Si todas las actividades de un trabajo están *finalizadas* → el trabajo pasa a *finalizado*.
   - Si todos los trabajos están *finalizados* → el proyecto pasa a *finalizado*.
3. **Restricciones de integridad:**
   - No se puede eliminar un proyecto con trabajos asociados (salvo en estado borrador).
   - No se puede cerrar un proyecto si hay trabajos en curso.
4. **Control de fechas (opcional):**
   - Las fechas de un trabajo deben estar dentro del rango de fechas del proyecto.
   - Las fechas de una actividad deben estar dentro del rango de fechas de su trabajo.

4. Funcionalidades deseables (versión ampliada)

- Filtros por estado, responsable, prioridad y fechas.
- Define en el sistema perfiles como Analista, jefe de proyecto, Desarrollador Junior, Desarrollador Senior y añádelos como responsables de actividades o de proyecto.
- Notificaciones automáticas cuando se cambia de estado.
  - Aviso por email al desarrollador que se le asigna una actividad.

    - Aviso por email al jefe de proyecto cuando se finaliza una actividad.
- Posibilidad de añadir documentos adjuntos en el proyecto como por ejemplo el análisis de requisitos.
- Informe resumen del proyecto (costes, avances, incidencias). (Pendiente de ver en clase los informes)

Ejemplo de estructura de proyecto para los modelos

```
my_addons/
└── project_management/
    ├── __init__.py
    ├── __manifest__.py
    └── models/
        ├── __init__.py
        ├── estados_proyecto.py
        ├── proyecto.py
        ├── trabajo.py
        └── actividad.py
    └── views/
            ├── __init__.py
            ├── estados_proyecto_view.py
            ├── proyecto_view.py
            ├── trabajo_view.py
            └── actividad_view.py
```

5. Organización de archivos de proyecto

- El archivo manifest debe estar completo con toda la información incluida el autor del proyecto.
- Todos los modelos deben estar separados en archivos. (Ver estructura anterior)
- Todas las vistas de cada modelo deben estar separadas en archivos que le identifiquen. Por ejemplo: proyecto_views.xml, estados_views.xml, trabajo_views.xml y actividad_views.xml.
- Todas las vistas de lista y formulario deben estar personalizadas.
- Los menús estarán separados igualmente en un archivo menus.xml.
- Debes generar archivos de demo para los estados de los proyectos, de los trabajos y de las actividades.
- El módulo debe tener un icono personalizado.

6. Entrega

- Se debe entregar la carpeta del módulo en un zip.
- Se debe hacer una demo del funcionamiento con datos con sentido.