CreativeMinds AI

*Desarrollo de mi módulo para Odoo versión 17*

Heily Madelay Ajila Tandazo

**Índice**

**1. Introducción**

*- 1.1. Descripción general del módulo y objetivos*

*- 1.2. Problemática que resuelve y valor añadido*

**2. Guía de Instalación**

*- 2.1. Requisitos previos y procesos de instalación paso a paso*

*- 2.2. Configuración inicial y verificación*

**3. Documentación Técnica**

*- 3.1. Arquitectura del módulo y estructura de archivos y directorios*

*- 3.2. Modelos y vistas*

*- 3.3. Seguridad y permisos*

**4. Manual de Usuario**

*- 4.1. Navegación por el módulo y funciones principales*

*- 4.2. Gestión (Proyectos, Empleados, Equipos)*

*- 4.3. Panel de control y solución de problemas frecuentes*

**5. Pruebas y Validación**

- *5.1. Pruebas realizadas*

*- 5.2. Resultados y validación*

**6. Conclusiones y Trabajo Futuro**

*- 6.1. Logros y limitaciones actuales*

*- 6.2. Propuestas de mejora a futuro*

**7. Referencias**

- *7.1. Referencias y recursos externos utilizados*

*- 7.2. Código fuente comentado y capturas de pantalla*

**1.Introducción**

**1.1. Descripción general del módulo y objetivos**

CreativeMinds AI es un módulo desarrollado para Odoo que integra herramientas de gestión de proyectos con capacidades de inteligencia artificial. Este módulo está diseñado específicamente para empresas que buscan optimizar la gestión de sus proyectos creativos y equipos de trabajo, proporcionando una interfaz intuitiva y funcionalidades avanzadas que facilitan la organización, seguimiento y control de las actividades del proyecto.

El módulo se estructura en cuatro componentes principales:

* Gestión de Proyectos
* Gestión de Empleados
* Gestión de Equipos
* Panel de Control

Tiene como objetivos principales:

* Facilitar la gestión integral de proyectos creativos
* Optimizar la asignación y seguimiento de recursos humanos
* Mejorar la colaboración entre equipos de trabajo
* Proporcionar métricas y análisis

Para alcanzar ser un sistema completo de gestión de proyectos acompañado de unas herramientas de seguimiento y control para la gestión de recursos humanos y equipos a través del análisis y reportes de rendimiento.

**1.2. Problemática que resuelve y el valor añadido**

El módulo CreativeMinds AI aborda varios desafíos comunes en la gestión de proyectos creativos:

1. **Desorganización en la gestión de proyectos:**
   * Dificultad para seguir el progreso de múltiples proyectos
   * Falta de centralización de la información
   * Comunicación ineficiente entre equipos
2. **Ineficiencia en la asignación de recursos:**
   * Sobrecarga o subutilización de personal
   * Dificultad en la planificación de capacidades
   * Falta de visibilidad sobre la disponibilidad de recursos
3. **Seguimiento inadecuado del progreso:**
   * Ausencia de métricas claras
   * Dificultad para identificar cuellos de botella
   * Falta de transparencia en el avance del proyecto

Con este módulo aportamos el valor significativo a las organizaciones a través de la mejora en la eficiencia operativa con un mayor de control y visibilidad gracias a los dashboards en tiempo real para que los equipos puedan realizar un seguimiento detallado de proyectos para mejorar en la toma de decisiones

**2. Guía de Instalación**

**2. 2. 1. Requisitos previos y procesos de instalación del módulo.**

Requisitos previos:

1. Tener instalado Odoo a partir de la versión 17.

2. Opcional: Tener Git instalado.

Proceso de instalación:

En caso de tener Git:

1.Abra git bash.

2.Cambia el directorio de trabajo actual a la ubicación en donde quieres clonar el repositorio (Directorio para módulos de tercero).

3.Escriba: git clone https://github.com/HeilyMadelay-hub/CreativeMinds-AI.git

4.Presione Entrar para crear el clon local .

En caso de no tener Git:

1.Entra en el repositorio: HeilyMadelay-hub/CreativeMinds-AI y bajar el .zip del repositorio.

2.Descomprimir repositorio e ir a la carpeta SourceCode.

Ahora hay dos opciones:

1. Copiar carpeta creativeminds a la carpeta de modulos de Odoo.

2. Crear un zip de la carpeta creativeminds, abrir Odoo y dar en “Importar Módulo” y elegir el zip.

**2.2. Configuración inicial y verificación.**

1. Modificar el archivo odoo.conf añadiendo el directorio en el que queramos trabajar con los nuevos módulos.

1. En Linux (Ubuntu, Debian, CentOS, etc.)

En sistemas basados en Linux, la ubicación más común del archivo odoo.conf es:

/etc/odoo/odoo.conf

/etc/odoo.conf (En algunos casos, el archivo puede estar directamente en esta ruta).

Si no se encuentra allí, puedes buscarlo con el comando:

2. En Windows

En Windows, la ubicación del archivo de configuración puede variar dependiendo de cómo hayas instalado Odoo. Sin embargo, algunas ubicaciones comunes son:

C:\Program Files\Odoo\server\odoo.conf

C:\Odoo\server\odoo.conf

Si no lo encuentras allí, puedes buscarlo en el directorio de instalación de Odoo.

3. En macOS

En macOS, si has instalado Odoo a través de un gestor de paquetes o mediante instalación manual, el archivo odoo.conf generalmente se encuentra en:

/etc/odoo.conf

/usr/local/odoo/odoo.conf

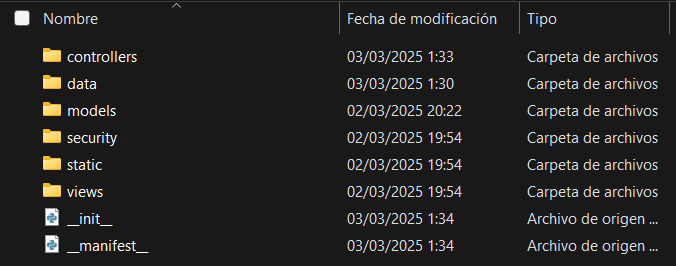
Creamos y/o accedemos al directorio, en este ejemplo se llama custom\_addons y fue creado dentro de la carpeta de odoo.

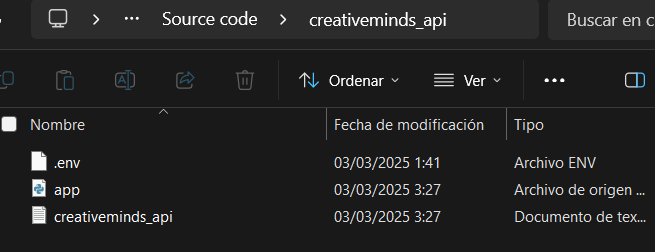
-Recuerda copiar el módulo desde la carpeta SourceCode o crear un zip del modulo

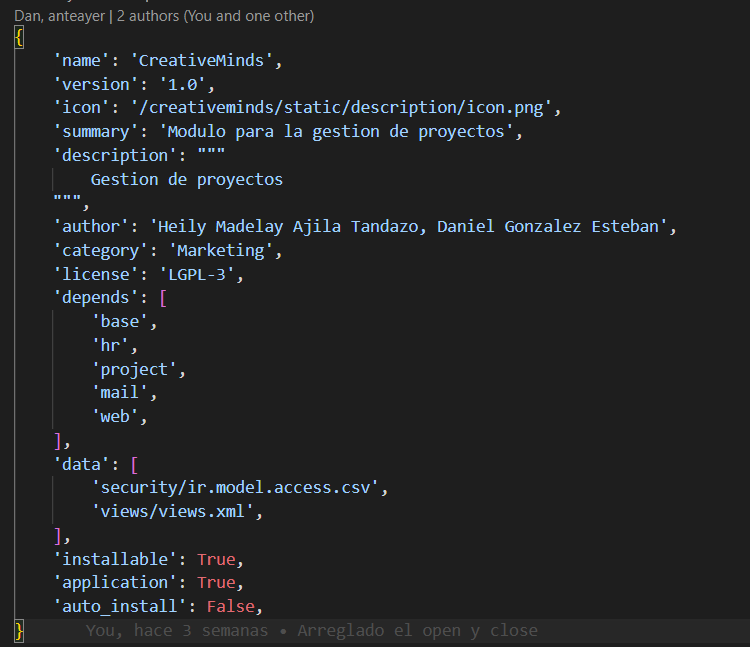
**3. Documentación Técnica**

**3. 1. Arquitectura del módulo y estructura de archivos y directorios.**

Este módulo sigue la arquitectura estándar de Odoo, que se organiza en una estructura modular con el mantenimiento de la extensibilidad. Siguiendo el patrón Modelo-Vista-Controlador, donde la lógica de negocio, la presentación y el control de flujo están claramente separados.





**Dependencias:**

El módulo depende de los siguientes módulos estándar de Odoo:

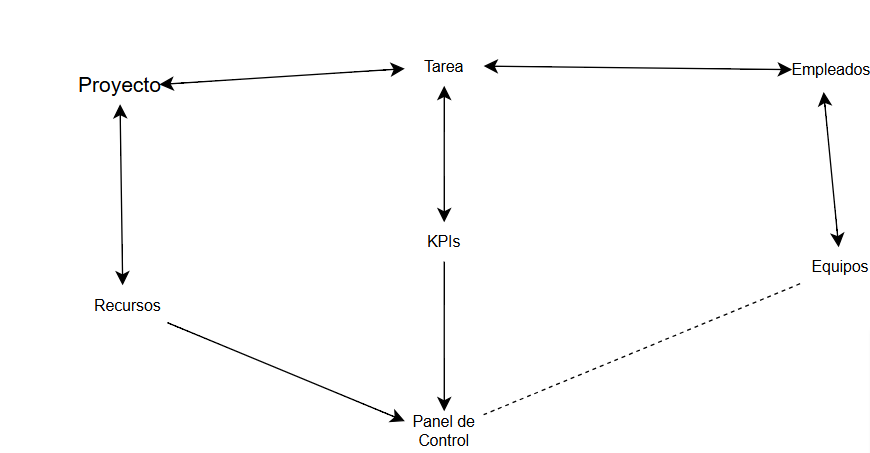
base: funcionalidades básicas de Odoo.

mail: sistema de comunicación y seguimiento

project: funcionalidades para la gestión de proyectos

hr: gestión de recursos humanos

**Flujo de datos:**

El diagrama de flujo de datos ilustra la estructura relacional del sistema CreativeMinds, donde las entidades principales —Proyectos, Tareas y Empleados— se interconectan bidireccionalmente, permitiendo que los proyectos contengan tareas, las tareas sean asignadas a empleados y los proyectos tengan personal asignado. Cada entidad principal se vincula con una entidad secundaria específica: los Proyectos se asocian con Recursos, las Tareas con KPIs y los Empleados con Equipos. Todas estas entidades secundarias convergen finalmente en el Panel de Control, que funciona como centro neurálgico para la visualización y monitorización integral de todos los componentes del sistema.

1. **2. Modelos y Vistas.**

**Modelos:**

**Vistas:**

**3. 3. Seguridad y permisos.**

La seguridad en el módulo se gestiona mediante un sistema de control de acceso basado en roles implementado a través de registros y definición de grupos.

* **Registros de Acceso (ir.model.access.csv):**

El módulo define permisos para cada modelo, estableciendo diferentes niveles de acceso según el rol del usuario en la organización:

* **Usuarios Básicos (base.group\_user):**
  + Tienen permisos para leer, crear y modificar registros, pero no pueden eliminarlos.
  + Este nivel de permisos se aplica a modelos operativos como Proyectos, Recursos, Tareas y KPIs.
  + Para modelos más sensibles como Empleados, Equipos, Horas Trabajadas y Panel de Control, únicamente se otorgan permisos de lectura.
* **Gestores de Proyecto (project.group\_project\_manager):**
  + Disponen de permisos completos (lectura, escritura, creación y eliminación) sobre la mayoría de los modelos.
  + Controlan Proyectos, Recursos, Tareas, KPIs, Equipos, Horas Trabajadas y Panel de Control.
* **Gestores de Recursos Humanos (hr.group\_hr\_manager):**
  + Tienen control completo sobre la información de Empleados.
  + Pueden crear, modificar y eliminar registros relacionados con el personal.
* **Grupos de Usuarios:**

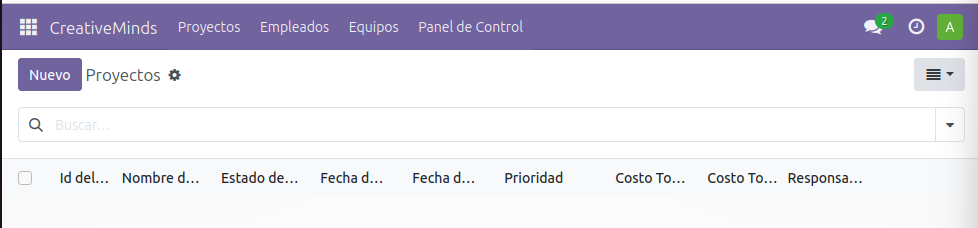
El sistema de seguridad se basa en tres grupos principales de Odoo:

* **base.group\_user:** Usuarios estándar de la plataforma. Acceden a información operativa del día a día con restricciones en operaciones críticas.
* **project.group\_project\_manager:** Responsables de la gestión de proyectos. Tienen control total sobre los recursos, tareas y elementos relacionados con proyectos.
* **hr.group\_hr\_manager:** Encargados de recursos humanos. Manejan la información sensible relacionada con empleados.

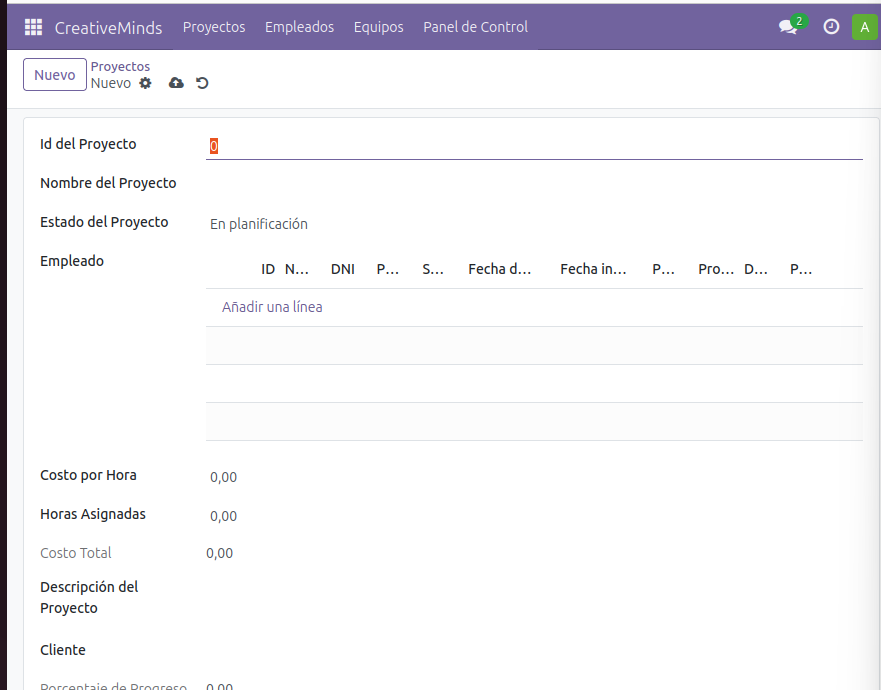
**4.Manual de Usuario**

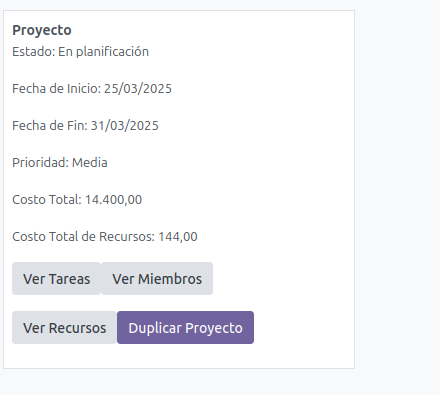
**4.1. Navegación por el módulo y funciones principales**

Nada mas entrar nos encontramos con “el protagonista del módulo”, Proyectos:

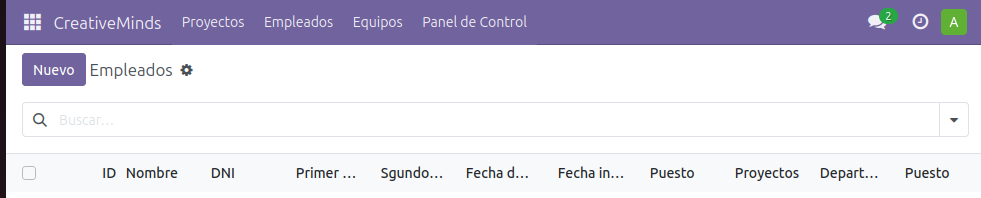
****

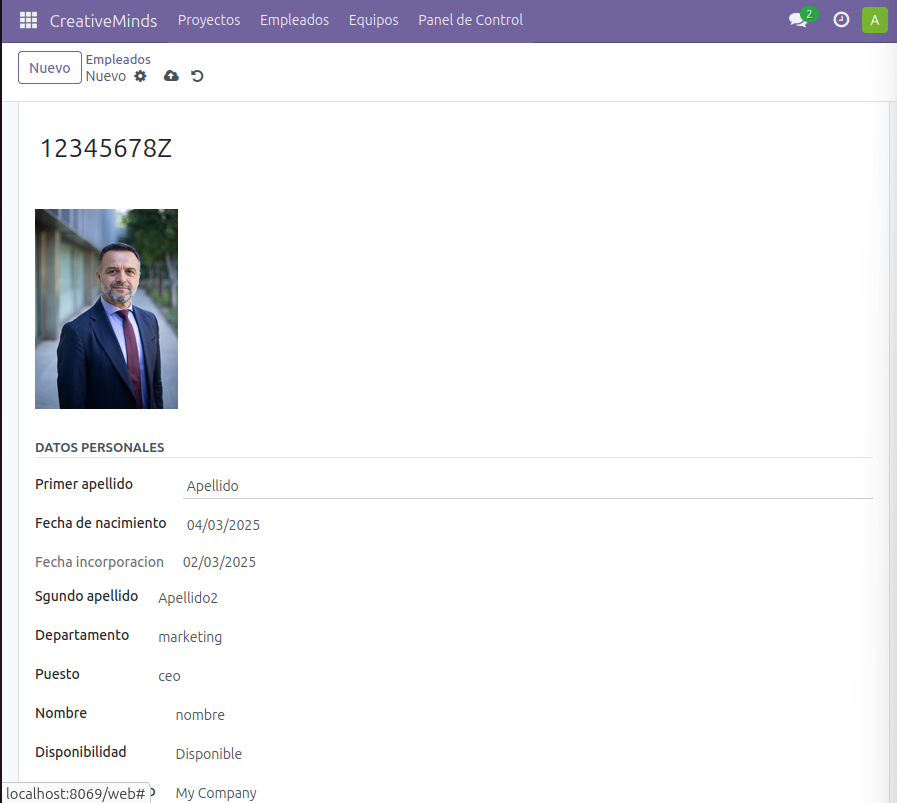
Vamos a proceder a crear un proyecto:

****

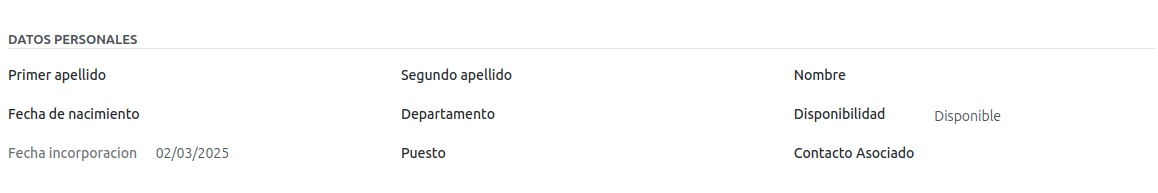
****

La clase Empleado, donde podremos crear un nuevo Empleado:

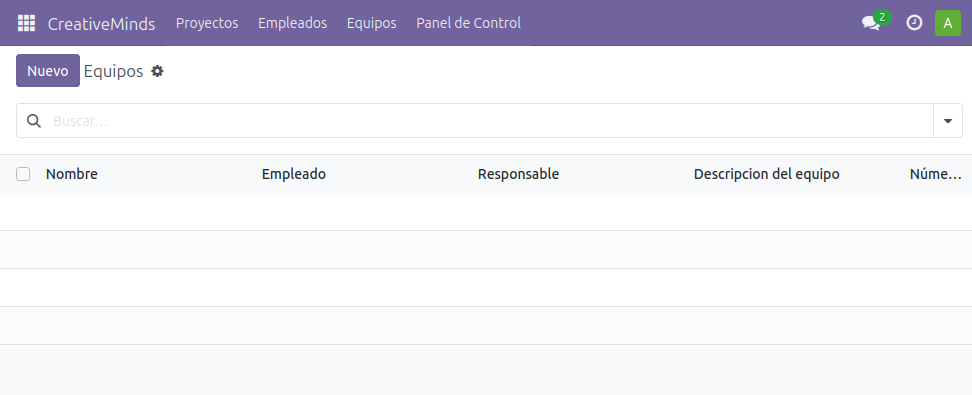


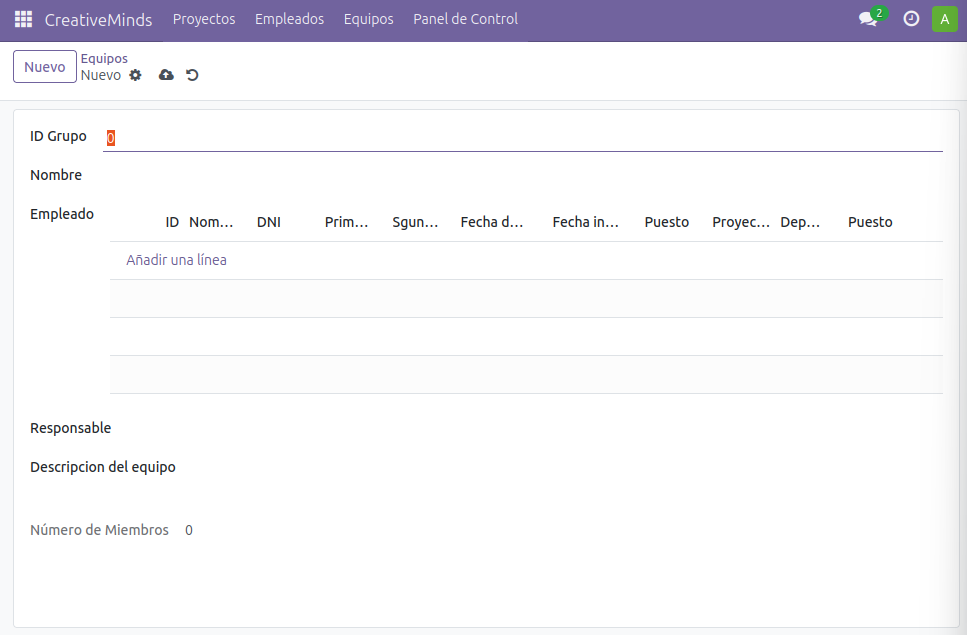


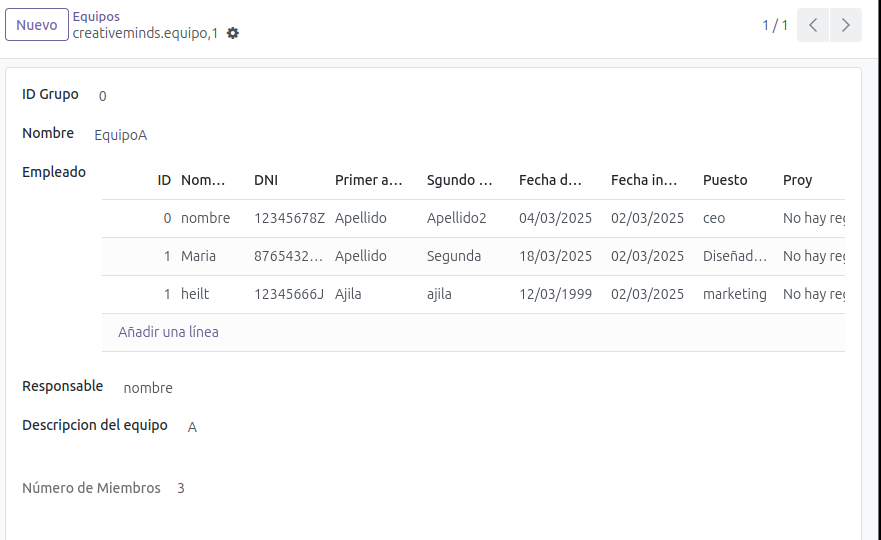
[El formulario corregido]:



La clase Equipos, donde podremos crear un nuevo Equipo para el proyecto:







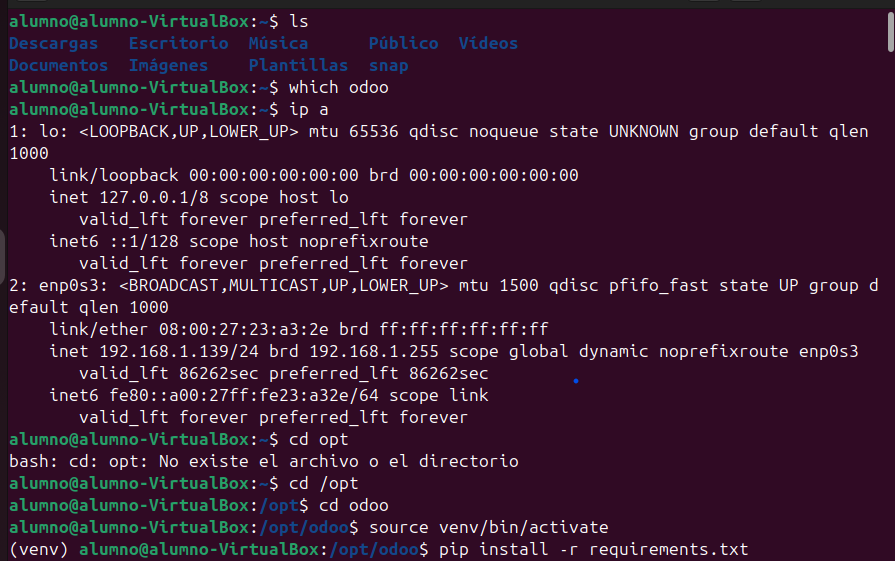
Creamos varios datos de Proyecto, Empleados y Equipos para pasar a mostrar Panel Control.

**4.2. Gestión (Proyectos, Empleados, Equipos)**

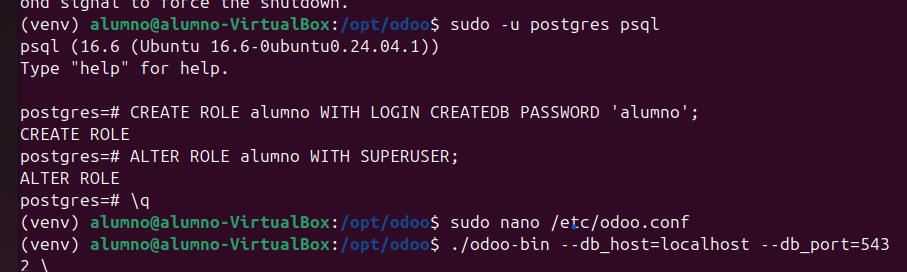
**4.3. Panel de control y solución de problemas frecuentes**

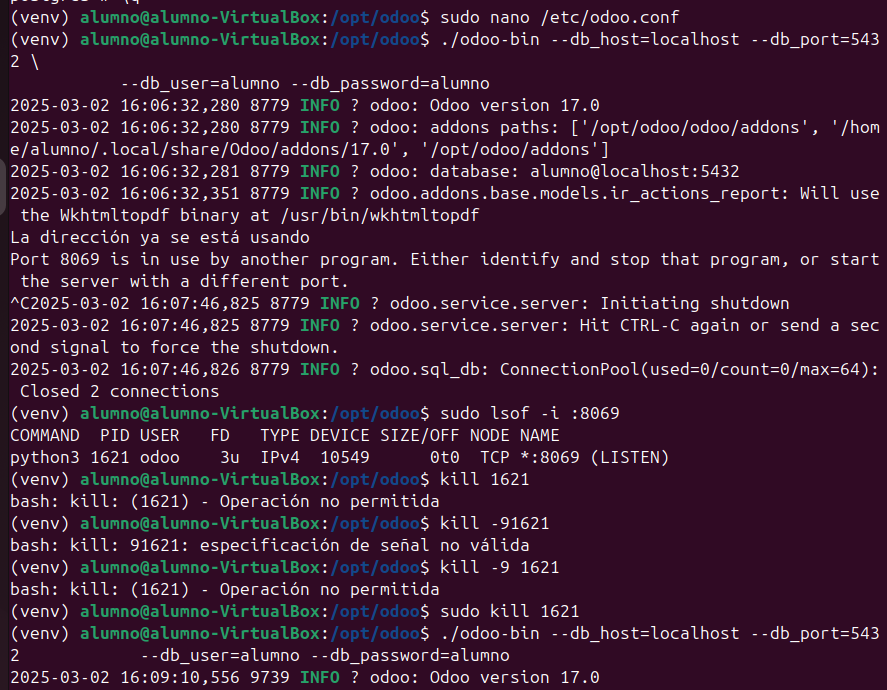
**5.Pruebas Realizadas y Resultados Validación**

Prueba de instalación:

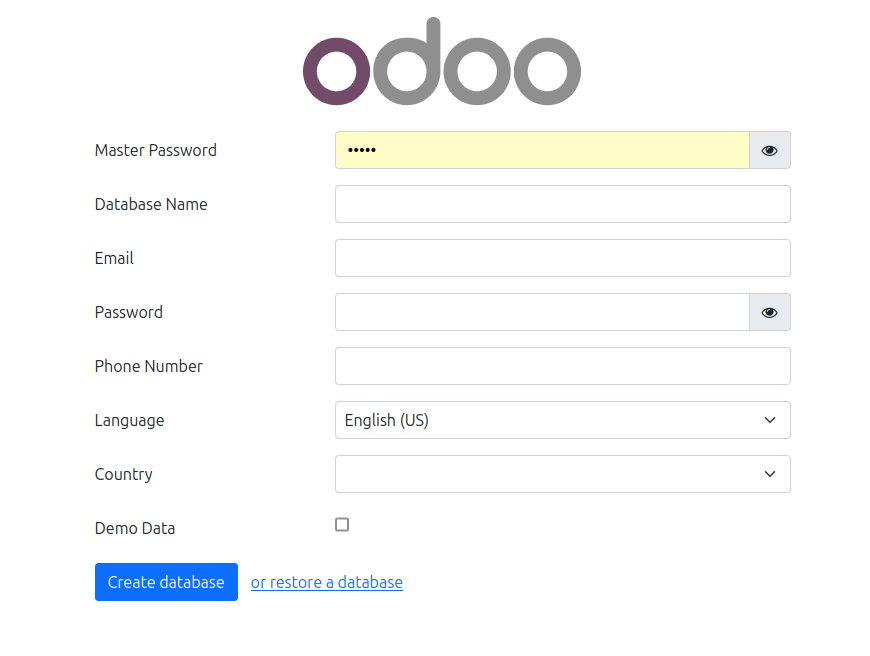


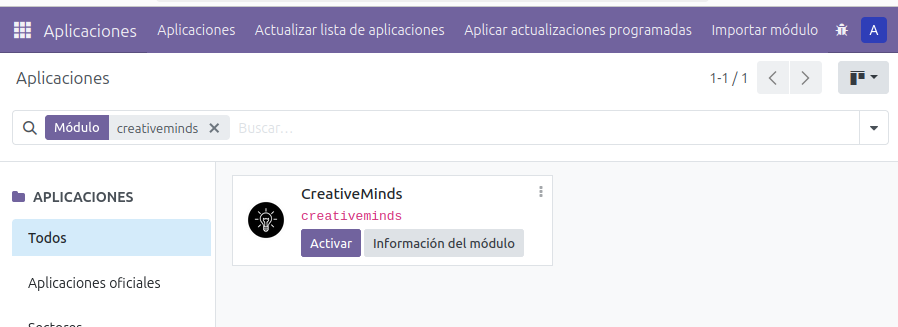


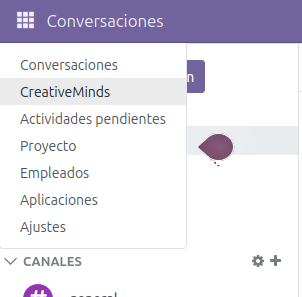




Luego después del último comando, nos mandara a la siguiente imagen para crear la base de datos:´

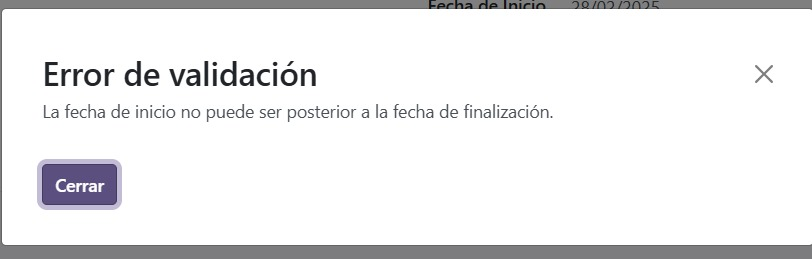


Vamos: Ajustes->Activamos el modo desarrollador->Aplicaciones->Actualizar lista de aplicaciones->Activar

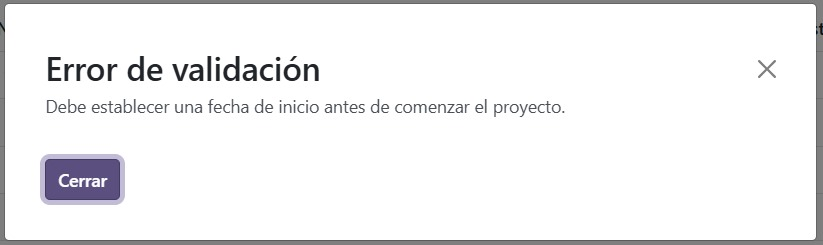
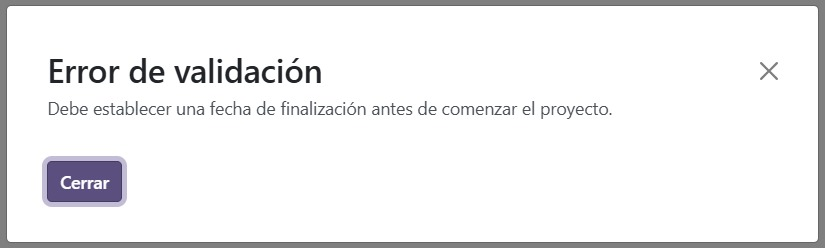


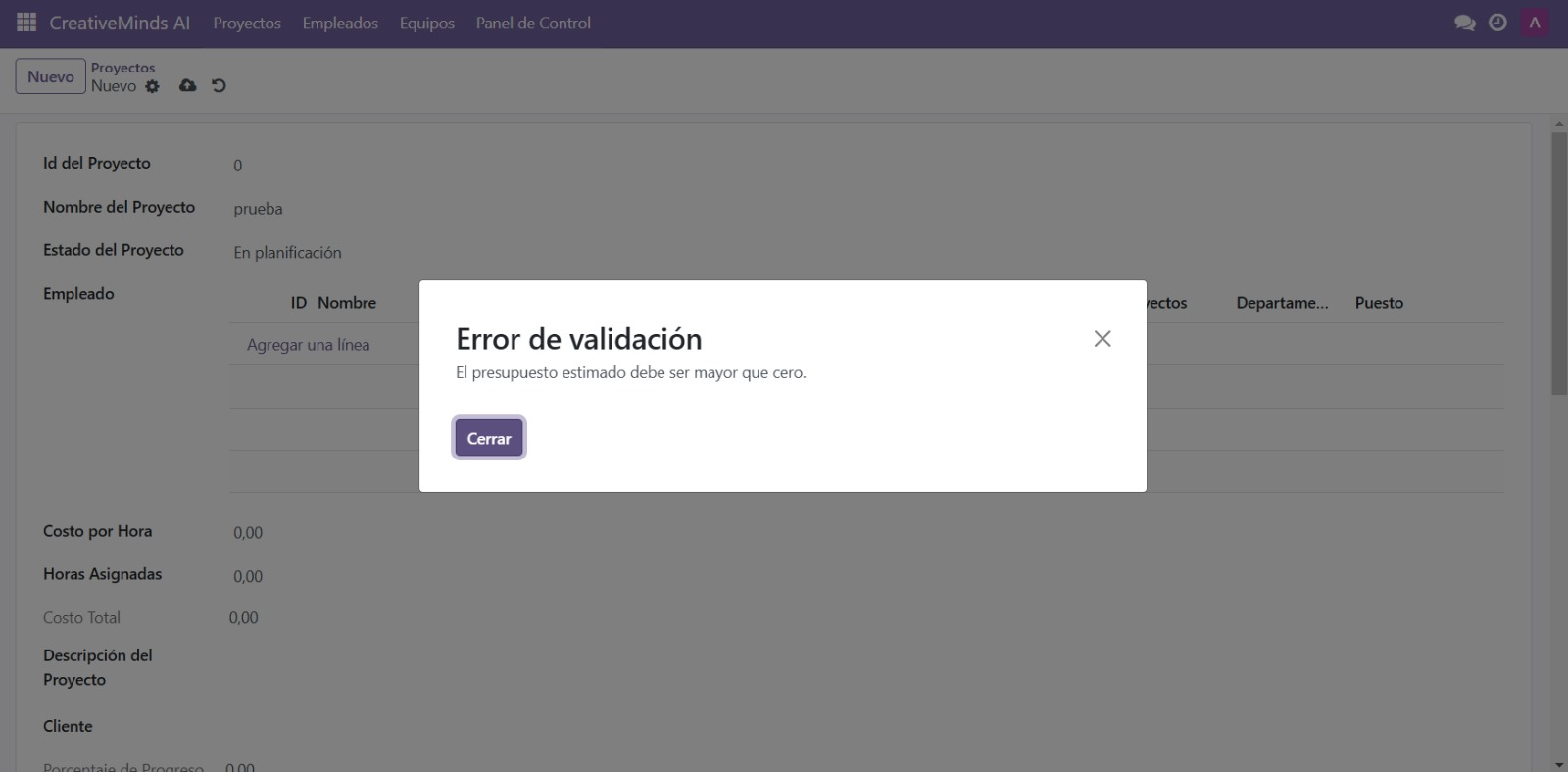
Prueba de validación de fechas

Caso 1: Intentamos crear un proyecto con una fecha de inicio posterior a la de finalización

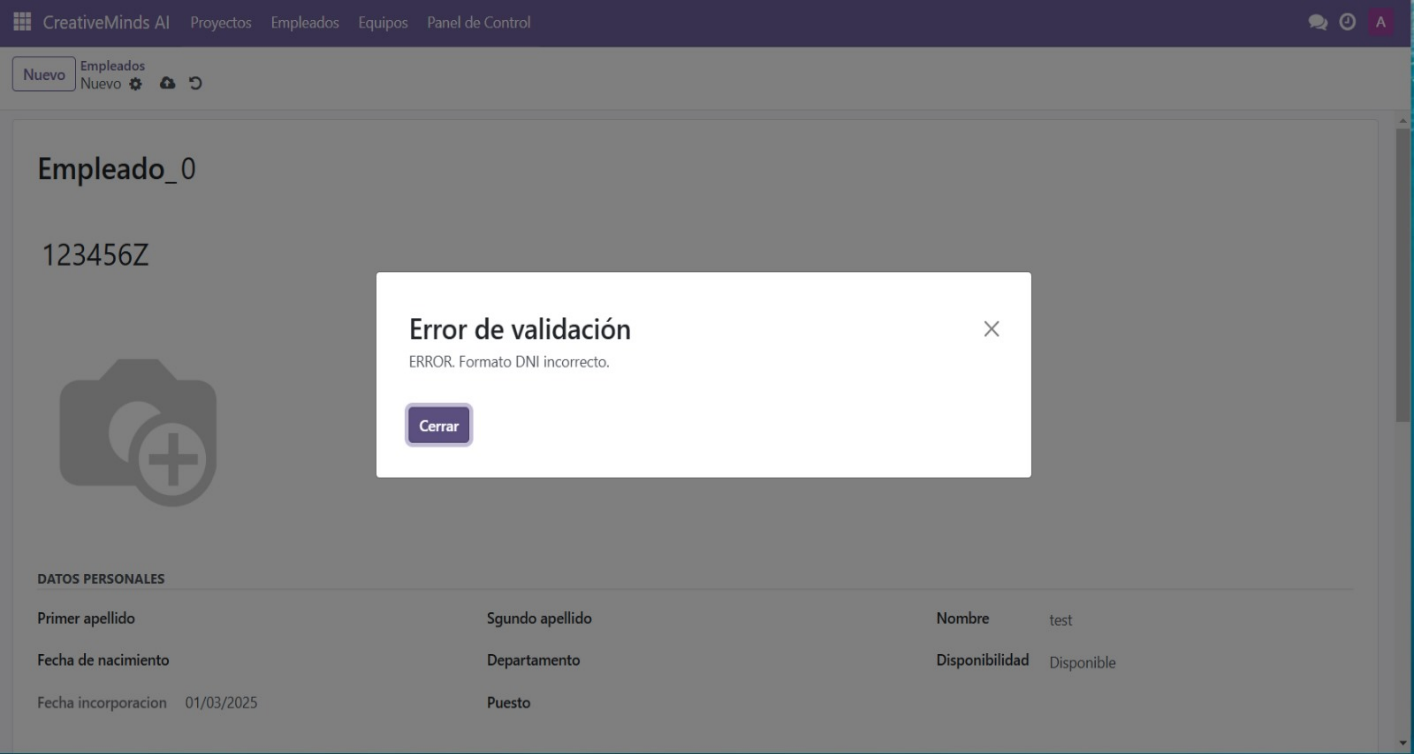


Caso 2: Intentamos crear un proyecto sin fecha de inicio/finalización.



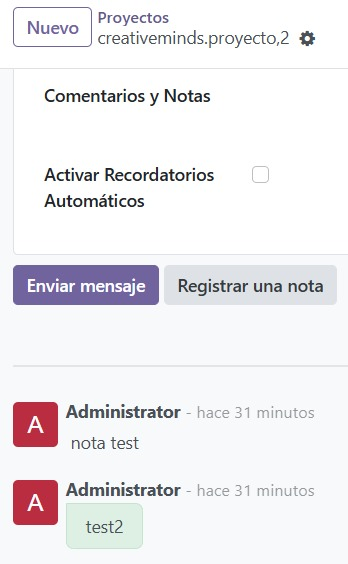
Prueba de validación de presupuesto

Prueba de validación de DNI

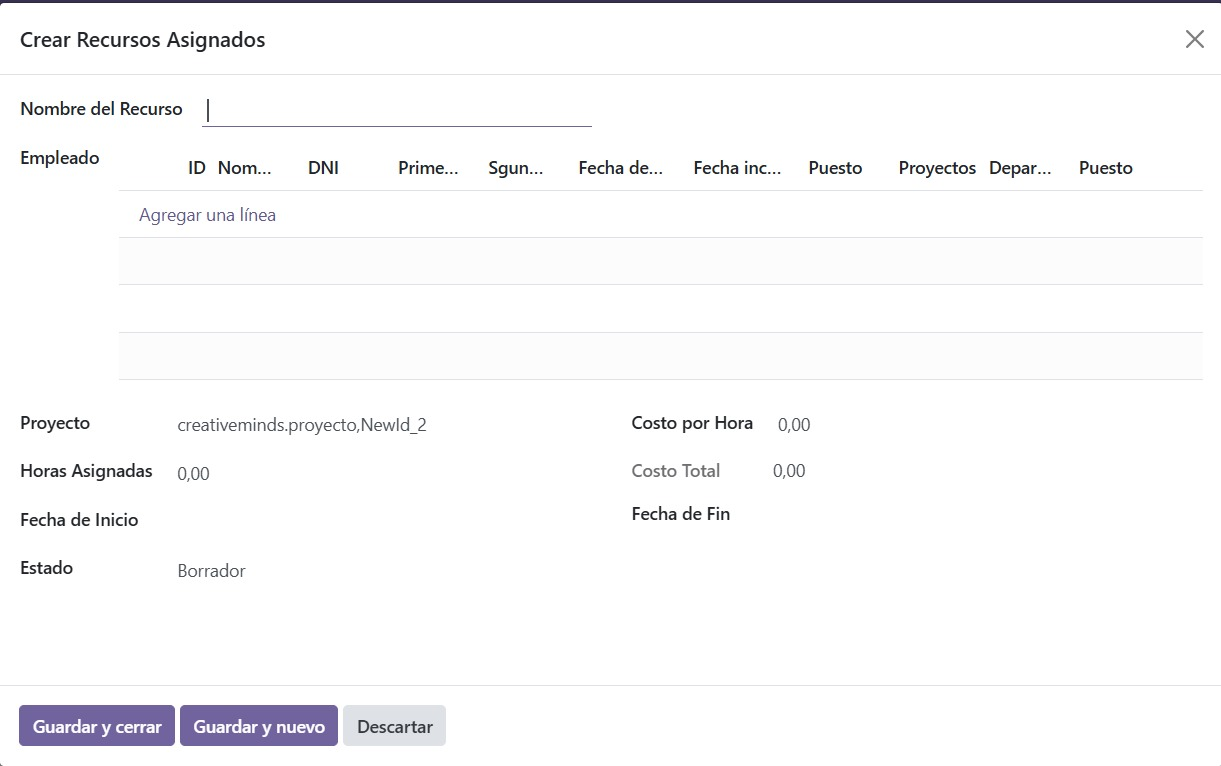
Caso: Creamos un empleado con un DNI incorrecto

Prueba de duplicación de proyecto

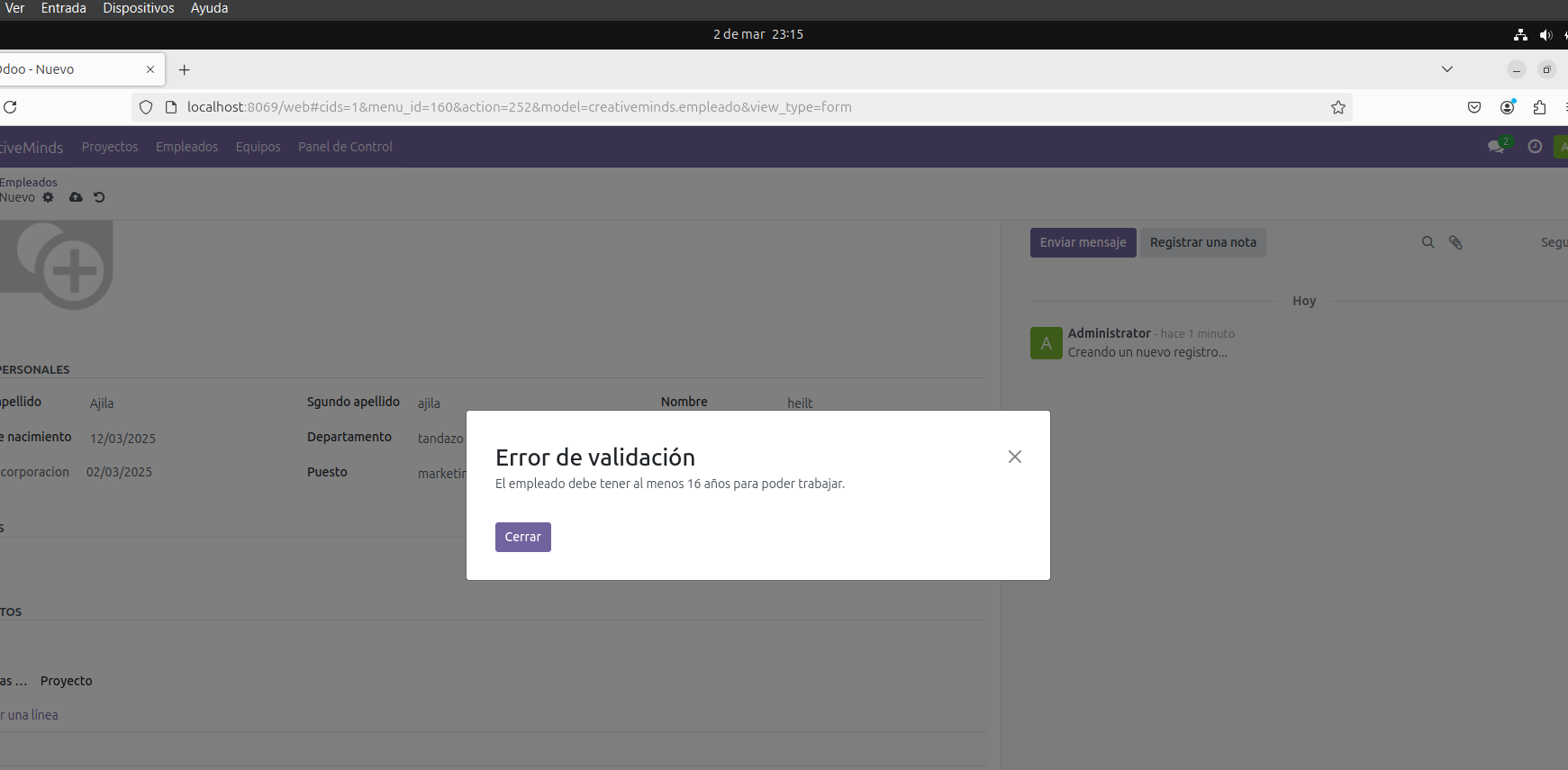
Prueba de envío de notificaciones:

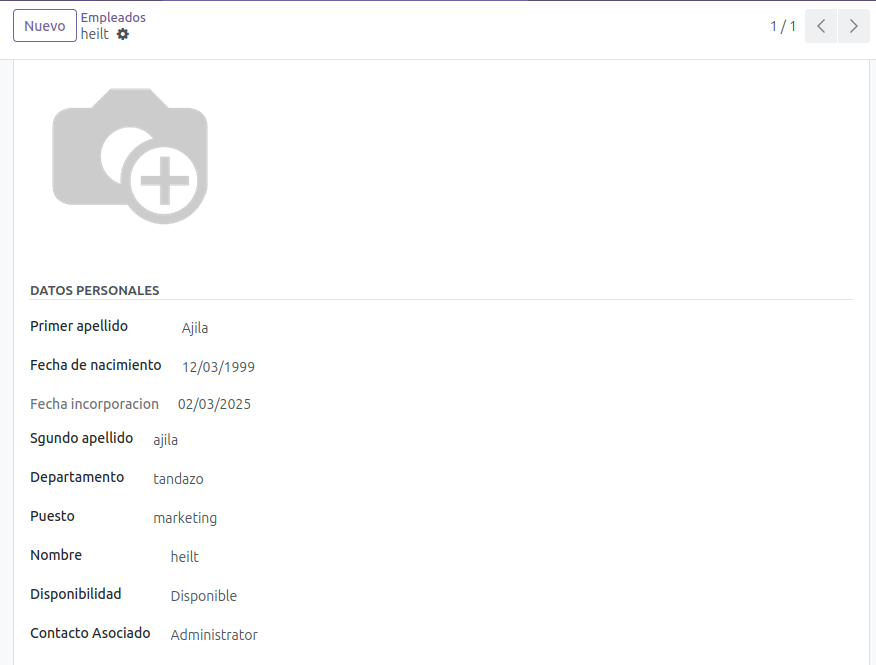


Prueba de creación de recursos:

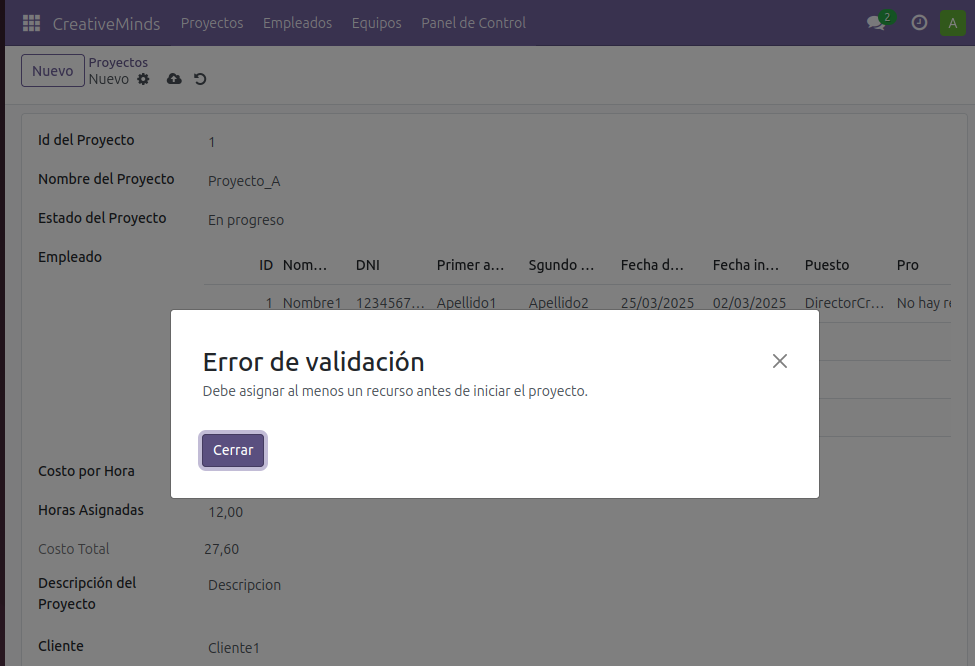


Prueba: No puede trabajar cualquier persona menos de 16 años:



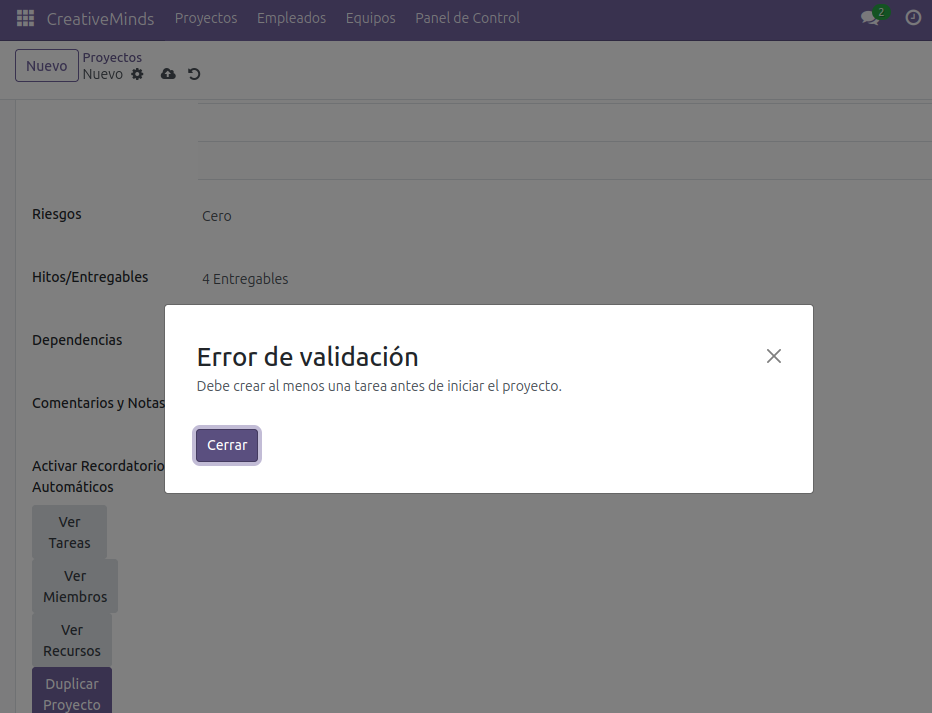


Prueba: Intentas crear un proyecto pero sin asignarle recursos:

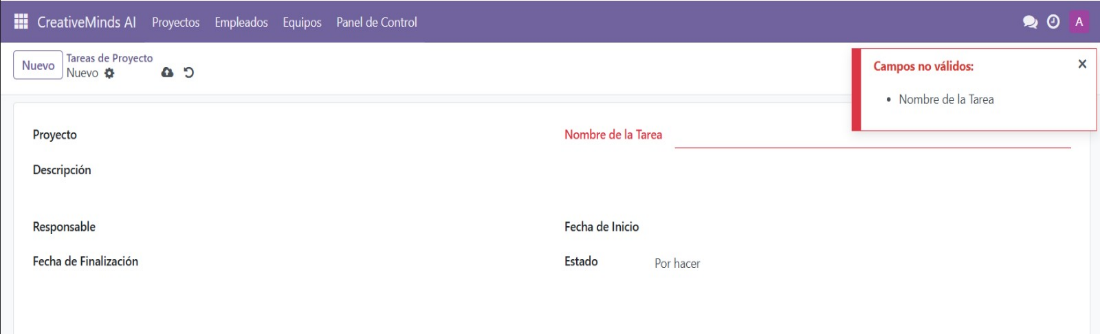


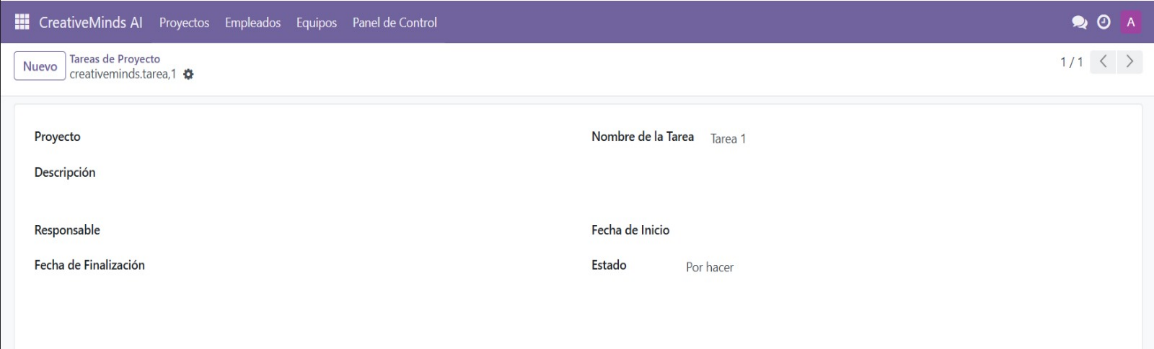
Prueba de creación de tareas:

Caso: Intentamos crear un proyecto sin tarea:



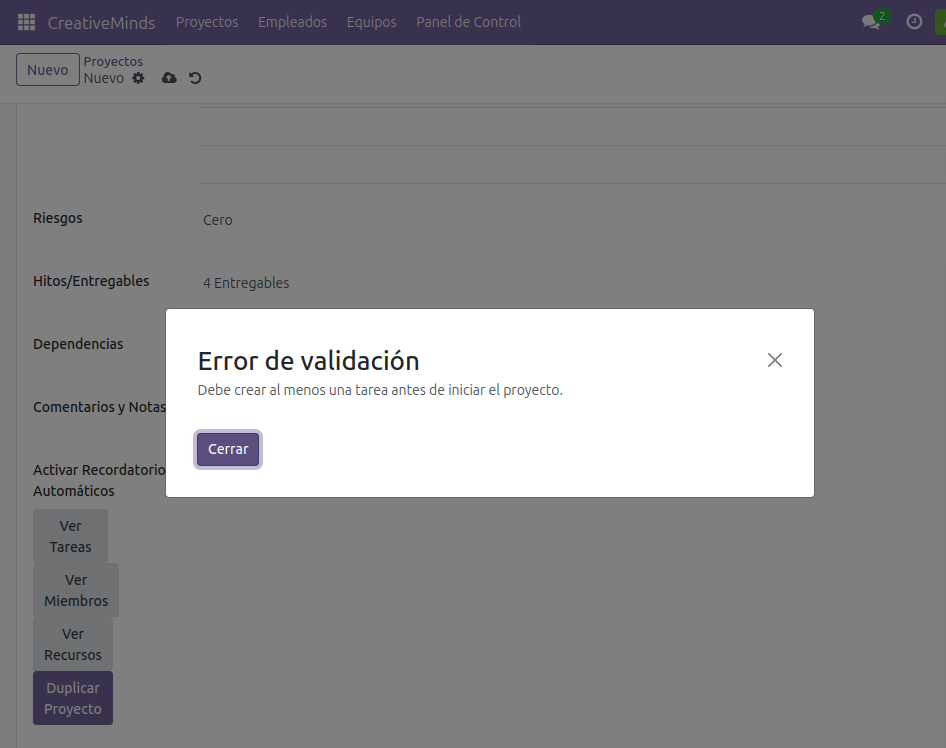
Caso: Intentamos crear una tarea sin nombre:



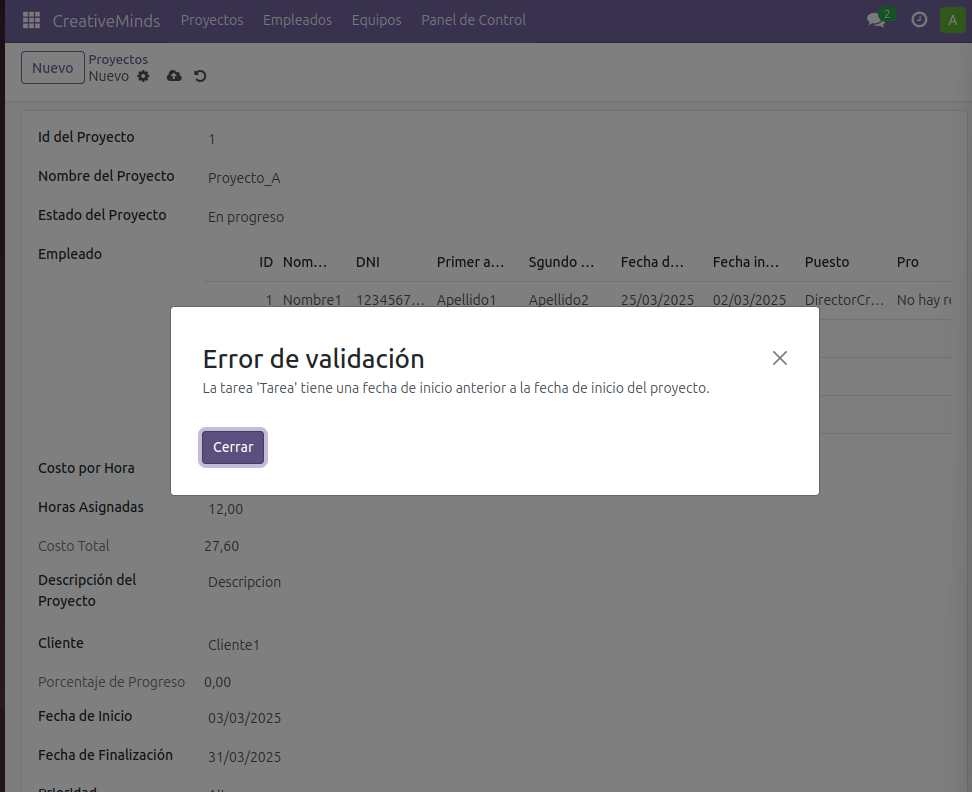


Una vez rellenado los datos, se ve así la tarea creada:

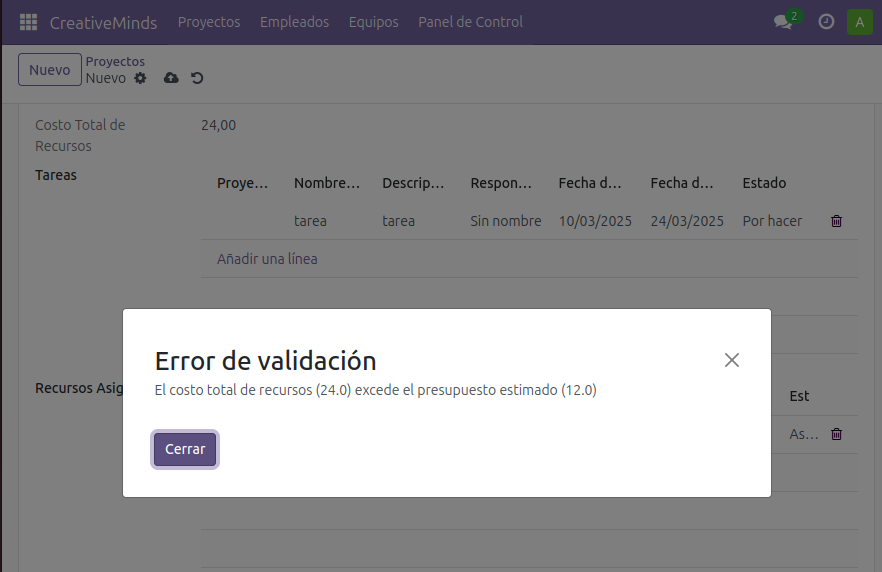
Pueba: Intentas ver los recursos y miembros sin haber inicializado tareas:



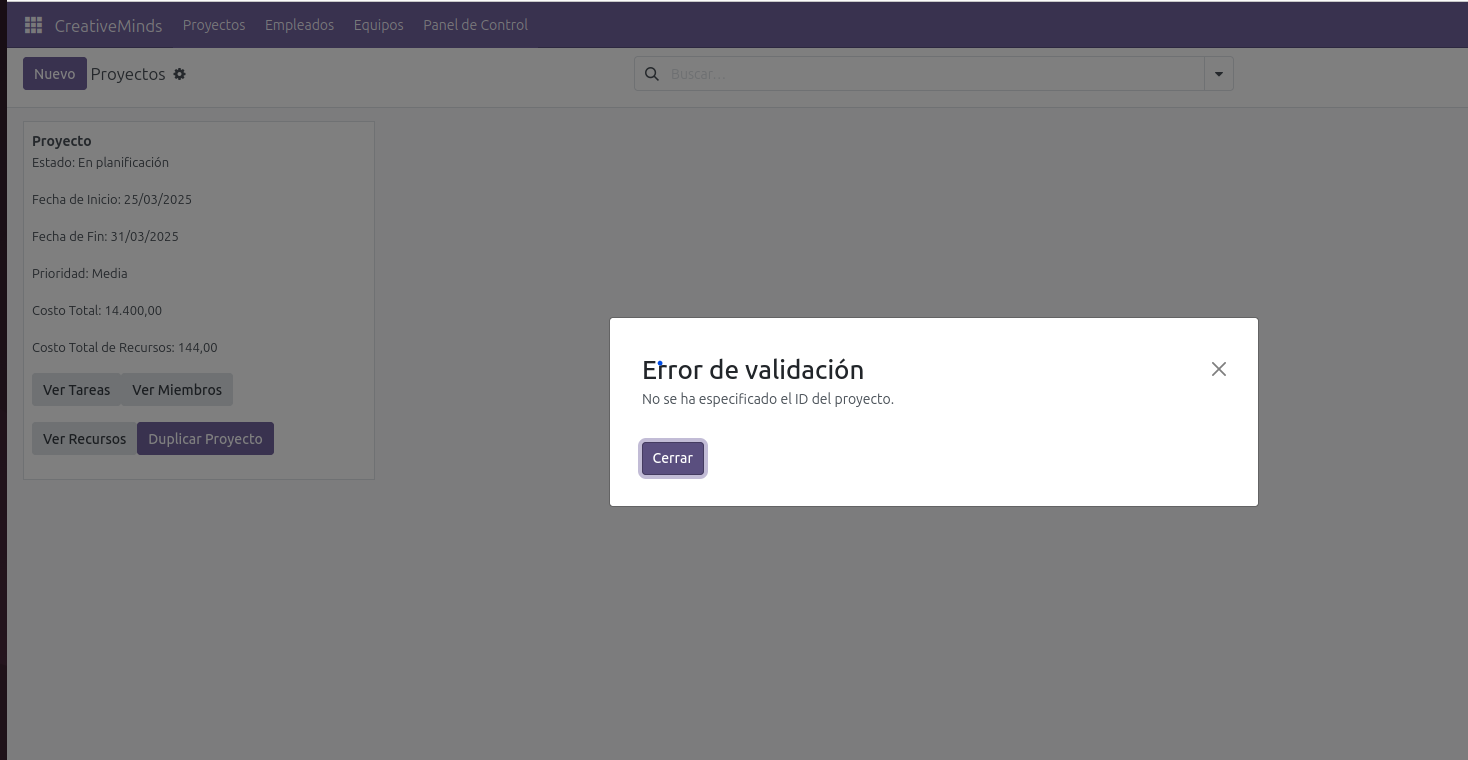
Prueba: Creas un proyecto con una tarea pero mal asignada la fecha:



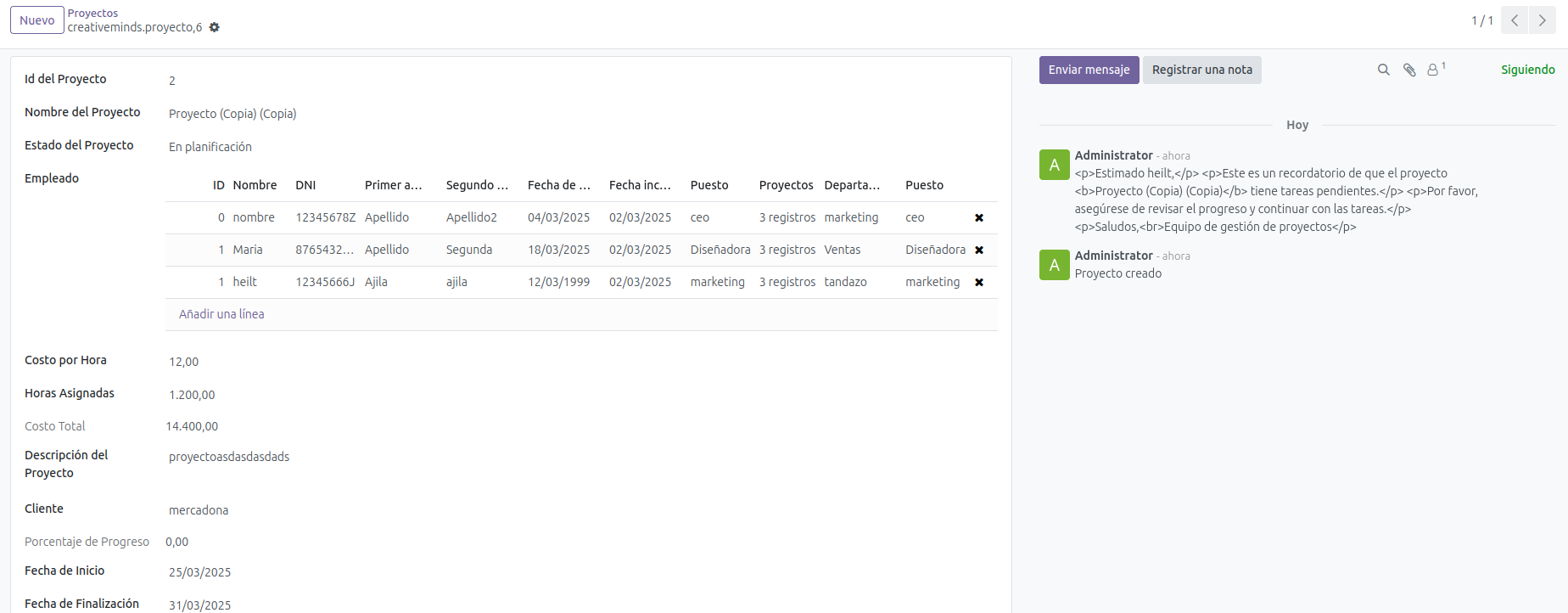
Prueba: Evitar que el usuario cree mal los recursos:

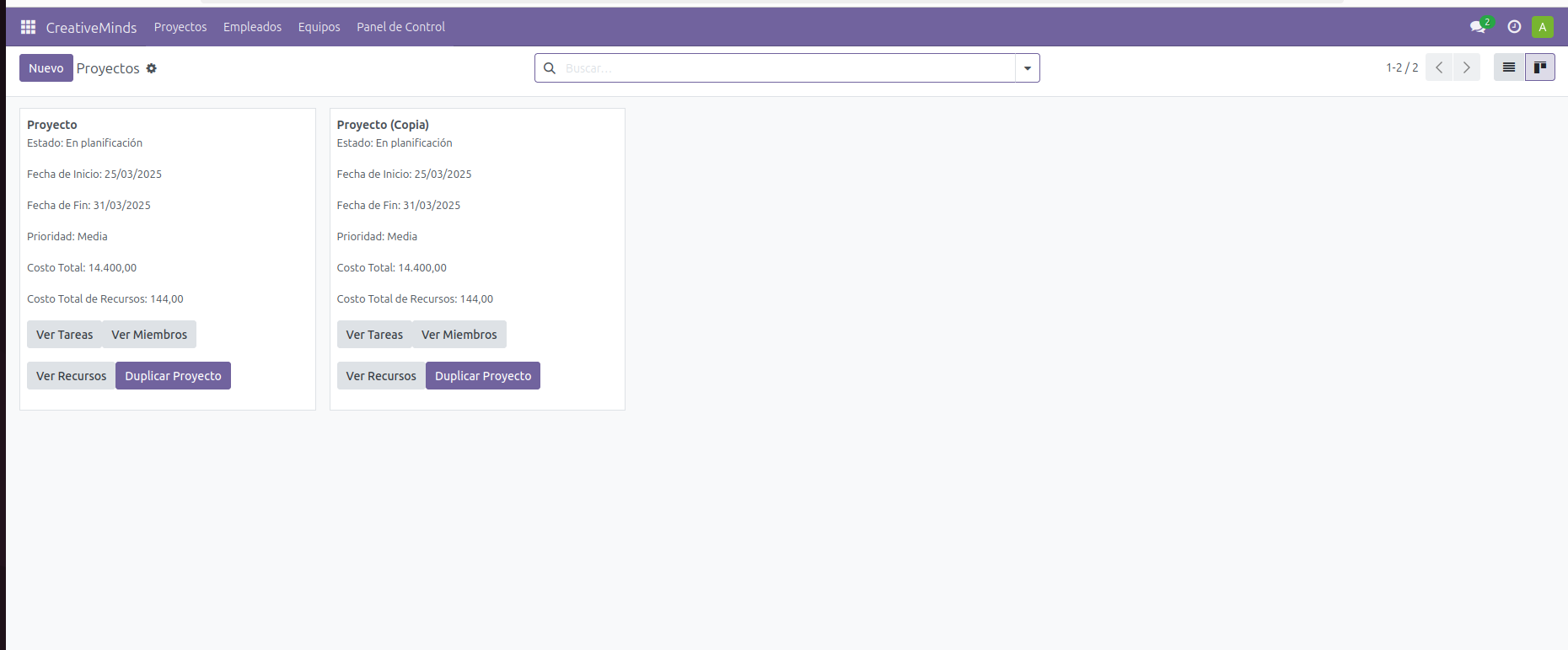


Caso: Intentas duplicar un proyecto sin especificar ID:



Caso: Duplicar el proyecto





**6. Conclusiones y Trabajo Futuro**

**6.1. Logros y limitaciones actuales**

|  |  |
| --- | --- |
| Logros | Limitaciones actuales |
|  | Dificultad para ser manejado por una persona que no este familiarizada con el área de gestión de proyectos |
|  | Falta de integración con otros módulos de Odoo como módulo Empleado |
|  | La API actual requiere conocimientos técnicos específicos que podrían dificultar su implementación por equipos sin experiencia en desarrollo. |
|  | Nos falta añadir más funcionalidades y botones al módulo como ‘Eliminar copia’ desde la vista panorámica en Proyectos |
|  |  |
|  |  |
|  |  |

**6.2. Propuestas de mejora a futuro**

1. Uso de Odoo Studio porque asi nos facilitaría la personalización de la interfaz y la configuración de flujos de trabajo sin necesidad de modificar código, lo que agiliza la adaptación del módulo a necesidades específicas.

2. Mejor Seguridad y Control de Accesos tras la implementación de un registro de auditoría permitirá monitorear cambios en la base de datos, mejorando la trazabilidad y el cumplimiento normativo.

3. Integración con Inteligencia Artificial (IA), para que permita potenciar diversas áreas de empresas, como el área de CRM y Marketing haciendo que puedan realizar un análisis predictivo para segmentar clientes y automatizar campañas junto con los chatbots que proporcionarían un mejor soporte inteligente

4. Mejora en Rendimiento y Escalabilidad, tras la optimización de la gestión de bases de datos, mejorando las consultas para que soporten múltiples servidores para una mejor escalabilidad en la nube.

5. Realizar una integración avanzada con herramientas de Business Intelligence (BI) como Power BI. Además, potenciar el desarrollo de nuestra APIs para hacerla más avanzadas permitirá una conexión fluida con otros sistemas empresariales. Implementa endpoints de exportación de datos compatibles con formatos estándar (JSON, CSV, Excel) y desarrolla una capa de abstracción con especificaciones OpenAPI, facilitando la integración directa con herramientas BI como Power BI y sistemas empresariales externos.

*Llegamos a esta conclusión tras analizar la estructura actual de la API de Creative Minds, que ya maneja datos analíticos complejos pero carece de métodos estandarizados de exportación. Observamos que implementar endpoints específicos para exportación en formatos universales (JSON, CSV, Excel) y documentar la API con especificaciones OpenAPI no solo facilitaría la integración con herramientas de BI como Power BI, sino que también permitiría a otros sistemas empresariales consumir estos datos de forma automatizada, multiplicando así el valor de la información ya procesada por nuestra plataforma.*

**7. Referencias**

**7.1. Referencias y recursos externos utilizados**

[Configuración del sistema — documentación de Odoo - 17.0](https://www.odoo.com/documentation/17.0/es/administration/on_premise/deploy.html)

[SGE-UT5-Desarrollo de módulos Odoo16 - YouTube](https://www.youtube.com/playlist?list=PLOgymZ7xUgbb4iTyZxCjAPO_lExJCpSXr)

[Welcome to Flask — Flask Documentation (3.1.x)](https://flask.palletsprojects.com/en/stable/)

[SQLAlchemy Documentation — SQLAlchemy 2.0 Documentation](https://docs.sqlalchemy.org/en/20/)

[Pandas DataFrames](https://www.w3schools.com/python/pandas/pandas_dataframes.asp#:~:text=Pandas%20DataFrames%20,table%20with%20rows%20and%20columns)

[How To Use a PostgreSQL Database in a Flask Application | DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-use-a-postgresql-database-in-a-flask-application)

[Tutorial Flask - Lección 5: Base de datos con Flask SQLAlchemy](https://j2logo.com/tutorial-flask-leccion-5-base-de-datos-con-flask-sqlalchemy/#:~:text=app%20%3D%20Flask%28__name__%29%20app.config,SQLALCHEMY_TRACK_MODIFICATIONS%27%5D%20%3D%20False)

https://www.odoo.com/documentation/15.0/es/applications/studio/fields.html

**7.2. Código fuente comentado y capturas de pantalla**

**Models:**

from odoo import models, fields, api # Importa los módulos necesarios de Odoo para la creación de modelos y campos.

from odoo.exceptions import ValidationError # Importa la excepción ValidationError para manejar errores de validación.

from datetime import date # Importa el módulo date para trabajar con fechas.

import re # Importa el módulo re para trabajar con expresiones regulares.

class Proyecto(models.Model):

\_name = 'creativeminds.proyecto' # Nombre técnico del modelo en Odoo.

\_description = 'Proyecto' # Descripción del modelo.

\_inherit = ['mail.thread', 'mail.activity.mixin'] # Herencia de funcionalidades de seguimiento de mensajes (para notificaciones y conversaciones).

# Campos básicos del proyecto

proyecto\_id = fields.Integer(string='Id del Proyecto', required=True) # Identificador único del proyecto

nombre = fields.Char(string='Nombre del Proyecto', required=True) # Nombre del proyecto

empleado\_id = fields.Many2many('creativeminds.empleado', string='Empleado') # Empleados asignados al proyecto

costo\_por\_hora = fields.Float(string='Costo por Hora') # Costo por hora de trabajo en el proyecto

horas\_asignadas = fields.Float(string='Horas Asignadas') # Total de horas asignadas al proyecto

costo\_total = fields.Float(string='Costo Total', compute='\_calcular\_costo\_total', store=True) # Cálculo del costo total basado en las horas y costo por hora

descripcion = fields.Text(string='Descripción del Proyecto') # Descripción detallada del proyecto

cliente = fields.Char(string='Cliente') # Cliente asociado al proyecto

@api.depends('costo\_por\_hora', 'horas\_asignadas')

def \_calcular\_costo\_total(self):

# Calcula el costo total del proyecto como el producto de costo por hora y horas asignadas

self.costo\_total = self.costo\_por\_hora \* self.horas\_asignadas

@api.constrains('costo\_por\_hora', 'horas\_asignadas')

def \_verificar\_costo\_y\_horas(self):

# Verifica que el costo por hora y las horas asignadas sean valores positivos

if self.costo\_por\_hora < 0 or self.horas\_asignadas < 0:

raise ValidationError("El costo por hora y las horas asignadas deben ser valores positivos.")

# Campos relacionados con el estado y seguimiento del proyecto

estado = fields.Selection([ # Estado del proyecto

('planificacion', 'En planificación'),

('en\_progreso', 'En progreso'),

('finalizado', 'Finalizado'),

('detenido', 'Detenido'),

], string='Estado del Proyecto', default='planificacion', tracking=True)

porcentaje\_progreso = fields.Float( # Porcentaje de progreso calculado

string='Porcentaje de Progreso',

compute='\_calcular\_progreso',

store=True,

tracking=True

)

# Fechas del proyecto

fecha\_inicio = fields.Date(string='Fecha de Inicio') # Fecha de inicio del proyecto

fecha\_fin = fields.Date(string='Fecha de Finalización') # Fecha de finalización del proyecto

# Prioridad y responsables

prioridad = fields.Selection([ # Nivel de prioridad del proyecto

('baja', 'Baja'),

('media', 'Media'),

('alta', 'Alta'),

], string='Prioridad', default='media')

responsable\_id = fields.Many2one('creativeminds.empleado', string='Responsable') # Responsable principal del proyecto

# Presupuesto y recursos

presupuesto\_estimado = fields.Float(string='Presupuesto Estimado') # Presupuesto estimado del proyecto

recursos\_ids = fields.One2many('creativeminds.recurso', 'proyecto\_id', string='Recursos Asignados') # Recursos asignados al proyecto

costo\_total\_recursos = fields.Float(string='Costo Total de Recursos',

compute='\_calcular\_costo\_total\_recursos',

store=True) # Costo total de los recursos asignados

@api.depends('recursos\_ids.costo\_total')

def \_calcular\_costo\_total\_recursos(self):

# Calcula el costo total de los recursos asignados al proyecto

self.costo\_total\_recursos = sum(self.recursos\_ids.mapped('costo\_total'))

@api.constrains('presupuesto\_estimado', 'costo\_total\_recursos')

def \_verificar\_presupuesto(self):

# Verifica que el costo total de los recursos no exceda el presupuesto estimado

if self.costo\_total\_recursos > self.presupuesto\_estimado:

raise ValidationError(

f"El costo total de recursos ({self.costo\_total\_recursos}) "

f"excede el presupuesto estimado ({self.presupuesto\_estimado})"

)

# Relaciones con tareas e indicadores

tareas\_ids = fields.One2many('creativeminds.tarea', 'proyecto\_id', string='Tareas') # Tareas asociadas al proyecto

indicadores\_ids = fields.One2many('creativeminds.kpi', 'proyecto\_id', string='Indicadores de Desempeño') # Indicadores de desempeño asociados al proyecto

# Archivos y documentación

imagen\_proyecto = fields.Binary(string='Imagen del Proyecto', attachment=True) # Imagen del proyecto

imagen\_filename = fields.Char(string='Nombre del archivo de imagen') # Nombre del archivo de imagen

documentacion\_tecnica = fields.Binary(string='Documentación Técnica', attachment=True) # Documentación técnica del proyecto

documentacion\_filename = fields.Char(string='Nombre del archivo de documentación') # Nombre del archivo de documentación

# Archivos adicionales

archivos\_adicionales = fields.Many2many(

'ir.attachment',

'creativeminds\_attachment\_rel',

'creativeminds\_id',

'attachment\_id',

string='Archivos Adicionales'

) # Archivos adicionales que se pueden asociar al proyecto

# Campos de texto para detalles adicionales

colaboradores = fields.Text(string='Agencias Colaboradoras') # Agencias colaboradoras en el proyecto

riesgos = fields.Text(string='Riesgos') # Riesgos asociados al proyecto

hitos = fields.Text(string='Hitos/Entregables') # Hitos o entregables importantes del proyecto

dependencias = fields.Text(string='Dependencias') # Dependencias entre el proyecto y otros

comentarios = fields.Text(string='Comentarios y Notas') # Notas o comentarios sobre el proyecto

# Configuración para recordatorios automáticos

recordatorios\_automaticos = fields.Boolean(string='Activar Recordatorios Automáticos')

# Verificación de fechas del proyecto

@api.constrains('fecha\_inicio', 'fecha\_fin')

def \_verificar\_fechas\_proyecto(self):

# Asegura que la fecha de inicio no sea posterior a la fecha de finalización

if self.fecha\_inicio and self.fecha\_fin and self.fecha\_inicio > self.fecha\_fin:

raise ValidationError("La fecha de inicio no puede ser posterior a la fecha de finalización.")

@api.constrains('presupuesto\_estimado')

def \_verificar\_presupuesto\_estimado(self):

# Verifica que el presupuesto estimado sea válido y suficiente para los recursos y tareas

if self.presupuesto\_estimado <= 0:

raise ValidationError("El presupuesto estimado debe ser mayor que cero.")

num\_recursos = len(self.recursos\_ids)

num\_tareas = len(self.tareas\_ids)

presupuesto\_requerido = (num\_recursos \* 500) + (num\_tareas \* 200) # Estimación de presupuesto requerido

if (num\_recursos > 0 or num\_tareas > 0) and self.presupuesto\_estimado < presupuesto\_requerido:

raise ValidationError(

f"El presupuesto ({self.presupuesto\_estimado}) es insuficiente. "

f"Se requieren al menos {round(presupuesto\_requerido, 2)} para cubrir recursos y tareas."

)

# Sobrescribimos el método 'create' para agregar lógica adicional al crear un proyecto.

@api.model

def create(self, valores):

proyecto = super(Proyecto, self).create(valores) # Crear el proyecto como se haría normalmente.

# Si los recordatorios automáticos están activados y el proyecto tiene un responsable asignado,

# se crea una tarea inicial y se envía un recordatorio.

if proyecto.recordatorios\_automaticos and proyecto.responsable\_id:

self.env['creativeminds.tarea'].create({

'nombre': f'Tarea inicial de proyecto: {proyecto.nombre}',# Nombre de la tarea inicial.

'proyecto\_id': proyecto.id, # Asociamos la tarea al proyecto.

'responsable\_id': proyecto.responsable\_id.id, # Asignamos al responsable del proyecto.

'estado': 'pendiente', # La tarea está pendiente al principio.

})

self.\_enviar\_recordatorio(proyecto) # Enviamos un recordatorio al responsable.

return proyecto # Devolvemos el proyecto creado.

# Método que actualiza el indicador de progreso del proyecto.

def actualizar\_progreso\_indicador(self):

# Buscamos el indicador específico relacionado con el progreso del proyecto.

indicador = self.env['creativeminds.indicador'].search([

('proyecto\_id', '=', self.id),

('nombre', '=', 'Progreso del Proyecto')

], limit=1)

if indicador: # Si el indicador existe, actualizamos su valor.

indicador.write({'valor': self.porcentaje\_progreso})

return True

# Método que envía una notificación al responsable del proyecto sobre el cambio de estado o progreso.

def enviar\_notificacion\_proyecto(self, tipo\_notificacion='estado'):

if not self.responsable\_id: # Si no hay responsable, no se envía notificación.

return

# Determinamos el asunto de la notificación según el tipo.

asunto = f"Cambio de Estado: Proyecto {self.nombre}" if tipo\_notificacion == 'estado' else f"Actualización de Progreso: Proyecto {self.nombre}"

# Construimos el mensaje de notificación en formato HTML.

mensaje = f"<p>Hola {self.responsable\_id.name},</p>"

mensaje += f"<p>El estado del proyecto <b>{self.nombre}</b> ha cambiado a <b>{dict(self.\_fields['estado'].selection).get(self.estado)}</b>.</p>"

mensaje += f"<p>Progreso actual: {round(self.porcentaje\_progreso, 2)}%</p>"

# Enviamos el mensaje por correo utilizando el sistema de mensajes de Odoo.

self.message\_post(

body=mensaje,

subject=asunto,

partner\_ids=[self.responsable\_id.partner\_id.id] # Enviamos el mensaje al partner del responsable.

)

# Método que envía un recordatorio al responsable del proyecto sobre las tareas pendientes.

def enviar\_recordatorio(self, proyecto):

if not proyecto.responsable\_id: # Si no hay responsable asignado, no se envía el recordatorio.

return

# Construimos el mensaje de recordatorio en formato HTML.

mensaje = f"""

<p>Estimado {proyecto.responsable\_id.name},</p>

<p>Este es un recordatorio de que el proyecto <b>{proyecto.nombre}</b> tiene tareas pendientes.</p>

<p>Por favor, asegúrese de revisar el progreso y continuar con las tareas.</p>

<p>Saludos,<br>Equipo de gestión de proyectos</p>

"""

# Enviamos el recordatorio por mensaje en Odoo.

proyecto.message\_post(

body=mensaje,

subject=f"Recordatorio: Proyecto {proyecto.nombre} - Tareas pendientes",

partner\_ids=[proyecto.responsable\_id.partner\_id.id] # Enviamos el mensaje al partner del responsable.

)

# Creamos una actividad para el responsable para asegurarnos de que se realice el seguimiento de las tareas pendientes.

actividad\_tipo = self.env.ref('mail.mail\_activity\_data\_todo')

modelo\_id = self.env['ir.model'].\_get\_id('creativeminds.proyecto')

self.env['mail.activity'].create({

'activity\_type\_id': actividad\_tipo.id,

'res\_model\_id': modelo\_id,

'res\_id': proyecto.id,

'user\_id': proyecto.responsable\_id.id,

'summary': f"Recordatorio: {proyecto.nombre} - Tareas pendientes",

'note': f"Este es un recordatorio para que revises las tareas pendientes del proyecto {proyecto.nombre}.",

})

# Método que calcula el porcentaje de progreso del proyecto basado en las tareas completadas.

@api.depends('tareas\_ids.estado')

def \_calcular\_progreso(self):

total\_tareas = len(self.tareas\_ids) # Contamos el total de tareas.

tareas\_completadas = len(self.tareas\_ids.filtered(lambda t: t.estado == 'completada'))

# Calculamos el progreso como el porcentaje de tareas completadas.

self.porcentaje\_progreso = (tareas\_completadas / total\_tareas \* 100) if total\_tareas > 0 else 0.0

# Método que obtiene un resumen detallado del proyecto.

def obtener\_resumen\_proyecto(self):

self.ensure\_one() # Aseguramos que solo haya un registro.

return {

'nombre': self.nombre,

'estado': self.estado,

'progreso': self.porcentaje\_progreso,

'presupuesto': {

'estimado': self.presupuesto\_estimado,

'actual': self.costo\_total\_recursos,

'disponible': self.presupuesto\_estimado - self.costo\_total\_recursos

},

'tareas': {

'total': len(self.tareas\_ids),

'completadas': len(self.tareas\_ids.filtered(lambda t: t.estado == 'completada')),

'en\_progreso': len(self.tareas\_ids.filtered(lambda t: t.estado == 'en\_progreso')),

'pendientes': len(self.tareas\_ids.filtered(lambda t: t.estado == 'pendiente'))

}

}

# Método que duplica un proyecto existente, copiando también sus tareas asociadas.

def duplicar\_proyecto(self, proyecto\_id):

if not proyecto\_id:

raise ValidationError("No se ha especificado el ID del proyecto.")

proyecto = self.browse(proyecto\_id) # Buscamos el proyecto por su ID.

if not proyecto.exists(): # Si el proyecto no existe, lanzamos una excepción.

raise ValidationError("El proyecto especificado no existe.")

nuevo\_proyecto = proyecto.copy() # Creamos una copia del proyecto.

# Copiamos las tareas asociadas al proyecto.

for tarea in proyecto.tareas\_ids:

tarea.copy({'proyecto\_id': nuevo\_proyecto.id})

return nuevo\_proyecto # Devolvemos el nuevo proyecto duplicado.

# Método para verificar que la descripción, cliente y responsable sean válidos antes de hacer cambios.

@api.constrains('descripcion', 'cliente', 'responsable\_id')

def \_verificar\_campos\_importantes(self):

if self.descripcion and len(self.descripcion.strip()) < 10: # La descripción debe tener al menos 10 caracteres.

raise ValidationError("La descripción del proyecto debe tener al menos 10 caracteres.")

if self.estado in ['en\_progreso', 'finalizado'] and not self.cliente: # El cliente es obligatorio si el estado es 'en progreso' o 'finalizado'.

raise ValidationError("Debe especificar un cliente antes de cambiar el proyecto a 'En progreso' o 'Finalizado'.")

if self.estado != 'planificacion' and not self.responsable\_id: # El responsable es obligatorio si el proyecto no está en planificación.

raise ValidationError("Debe asignar un responsable antes de avanzar con el proyecto.")

# Método para verificar que los campos de planificación son correctos para proyectos de alta prioridad.

@api.constrains('riesgos', 'hitos')

def \_verificar\_campos\_planificacion(self):

if self.prioridad == 'alta': # Si el proyecto es de alta prioridad.

if not self.riesgos: # Los riesgos deben estar definidos.

raise ValidationError("Para proyectos de alta prioridad, es obligatorio definir los riesgos.")

if not self.hitos: # Los hitos deben estar definidos.

raise ValidationError("Para proyectos de alta prioridad, es obligatorio definir los hitos/entregables.")

# Método para verificar que haya al menos un recurso asignado antes de cambiar el estado a 'en progreso'.

@api.constrains('recursos\_ids')

def \_verificar\_recursos\_minimos(self):

if self.estado != 'planificacion' and not self.recursos\_ids: # Si el proyecto no está en planificación y no tiene recursos asignados.

raise ValidationError("Debe asignar al menos un recurso antes de iniciar el proyecto.")

# Método para verificar que las fechas de inicio y fin, así como las tareas, sean coherentes.

@api.constrains('fecha\_inicio', 'fecha\_fin', 'estado', 'tareas\_ids')

def \_verificar\_fechas\_y\_tareas(self):

if self.estado == 'en\_progreso': # Si el proyecto está en progreso.

if not self.fecha\_inicio: # La fecha de inicio debe estar definida.

raise ValidationError("Debe establecer una fecha de inicio antes de comenzar el proyecto.")

if not self.fecha\_fin: # La fecha de fin debe estar definida.

raise ValidationError("Debe establecer una fecha de finalización antes de comenzar el proyecto.")

if not self.tareas\_ids: # Deben existir al menos una tarea asociada.

raise ValidationError("Debe crear al menos una tarea antes de iniciar el proyecto.")

for tarea in self.tareas\_ids:

if tarea.fecha\_inicio and tarea.fecha\_inicio < self.fecha\_inicio: # La tarea no puede empezar antes de la fecha de inicio del proyecto.

raise ValidationError(f"La tarea '{tarea.nombre}' tiene una fecha de inicio anterior a la fecha de inicio del proyecto.")

if tarea.fecha\_fin and tarea.fecha\_fin > self.fecha\_fin: # La tarea no puede terminar después de la fecha de finalización del proyecto.

raise ValidationError(f"La tarea '{tarea.nombre}' tiene una fecha de finalización posterior a la fecha de fin del proyecto.")

# Función para ver las tareas del proyecto

def ver\_tareas(self):

return {

'name': 'Tareas del Proyecto',

'type': 'ir.actions.act\_window',

'res\_model': 'creativeminds.tarea',

'view\_mode': 'tree,form',

'domain': [('proyecto\_id', '=', self.id)],

'context': {'default\_proyecto\_id': self.id},

'target': 'current',

}

# Función para ver los recursos del proyecto

def ver\_recursos(self):

return {

'name': 'Recursos del Proyecto',

'type': 'ir.actions.act\_window',

'res\_model': 'creativeminds.recurso',

'view\_mode': 'tree,form',

'domain': [('proyecto\_id', '=', self.id)],

'context': {'default\_proyecto\_id': self.id},

'target': 'current',

}

# Función para ver los miembros del proyecto

def ver\_miembros(self):

return {

'name': 'Miembros del Proyecto',

'type': 'ir.actions.act\_window',

'res\_model': 'creativeminds.empleado',

'view\_mode': 'tree,form',

'domain': [('proyecto\_id', '=', self.id)],

'context': {'default\_proyecto\_id': self.id},

'target': 'current',

}

class Recurso(models.Model):

\_name = 'creativeminds.recurso'

\_description = 'Recursos del Proyecto'

# Campos básicos

nombre = fields.Char(string='Nombre del Recurso', required=True) # Nombre del recurso (obligatorio).

empleado\_id = fields.Many2many('creativeminds.empleado', string='Empleado') # Relación con los empleados asignados al recurso.

proyecto\_id = fields.Many2one('creativeminds.proyecto', string='Proyecto') # Relación con el proyecto al que pertenece el recurso.

# Costos y presupuesto

costo\_por\_hora = fields.Float(string='Costo por Hora') # Costo por hora del recurso.

horas\_asignadas = fields.Float(string='Horas Asignadas') # Número de horas asignadas al recurso.

costo\_total = fields.Float(string='Costo Total', compute='\_compute\_costo\_total', store=True) # Cálculo automático del costo total basado en el costo por hora y las horas asignadas.

@api.depends('costo\_por\_hora', 'horas\_asignadas') # Cuando cambian el costo o las horas, se recalcula el costo total.

def \_compute\_costo\_total(self):

for record in self:

record.costo\_total = record.costo\_por\_hora \* record.horas\_asignadas # El costo total es el costo por hora multiplicado por las horas asignadas.

# Fechas de asignación

fecha\_inicio = fields.Date(string='Fecha de Inicio') # Fecha de inicio de la asignación del recurso.

fecha\_fin = fields.Date(string='Fecha de Fin') # Fecha de finalización de la asignación del recurso.

# Estado del recurso

estado = fields.Selection([ # Definimos los posibles estados del recurso.

('borrador', 'Borrador'), # Estado inicial.

('asignado', 'Asignado'), # Cuando el recurso ya está asignado al proyecto.

('en\_progreso', 'En Progreso'), # Cuando el recurso está trabajando activamente en el proyecto.

('completado', 'Completado') # Cuando el recurso ha finalizado su tarea.

], string='Estado', default='borrador') # El estado por defecto es 'borrador'.

class Tarea(models.Model):

\_name = 'creativeminds.tarea'

\_description = 'Tareas del Proyecto'

# Campos básicos

proyecto\_id = fields.Many2one('creativeminds.proyecto', string='Proyecto') # Relación con el proyecto al que pertenece la tarea.

nombre = fields.Char(string='Nombre de la Tarea', required=True) # Nombre de la tarea (obligatorio).

descripcion = fields.Text(string='Descripción') # Descripción opcional de la tarea.

responsable\_id = fields.Many2one('creativeminds.empleado', string='Responsable') # Relación con el empleado que es responsable de la tarea.

fecha\_inicio = fields.Date(string='Fecha de Inicio') # Fecha en la que la tarea debería comenzar.

fecha\_fin = fields.Date(string='Fecha de Finalización') # Fecha en la que la tarea debe finalizar.

estado = fields.Selection([ # Selección de estados de la tarea.

('pendiente', 'Por hacer'), # Estado cuando la tarea aún no se ha comenzado.

('en\_progreso', 'En progreso'), # Estado cuando la tarea está siendo trabajada.

('completada', 'Completada'), # Estado cuando la tarea ha sido finalizada.

], string='Estado', default='pendiente') # El estado inicial es "pendiente" por defecto.

# Método de validación de fechas

@api.constrains('fecha\_inicio', 'fecha\_fin') # Este decorador valida las fechas de inicio y fin.

def \_verificar\_fechas\_tarea(self):

if self.fecha\_inicio and self.fecha\_fin and self.fecha\_inicio > self.fecha\_fin: # Si la fecha de inicio es mayor que la de fin, lanza un error.

raise ValidationError("La fecha de inicio no puede ser posterior a la fecha de finalización.") # Lanza un error de validación si las fechas no son correctas.

#Indicadores de Desempeño

class KPI(models.Model):

\_name = 'creativeminds.kpi'

\_description = 'Indicadores Clave de Rendimiento'

# Campos de datos

proyecto\_id = fields.Many2one('creativeminds.proyecto', string='Proyecto') # Relación con el proyecto al que pertenece el KPI.

nombre = fields.Char(string='Nombre del KPI', required=True) # Nombre del KPI, que es obligatorio.

valor = fields.Float(string='Valor') # Valor actual del KPI, que puede ser un número decimal.

objetivo = fields.Float(string='Objetivo') # Objetivo o meta del KPI, también como número decimal.

class Empleado(models.Model):

\_name = 'creativeminds.empleado'

\_description = 'Empleados del Proyecto'

\_inherit = ['mail.thread']

# Campos de datos

empleado\_id = fields.Integer(string='ID', required=True) # ID único del empleado, obligatorio.

nombre = fields.Char(string='Nombre', required=True) # Nombre del empleado, obligatorio.

dni = fields.Char(string='DNI', size=9, required=True) # DNI del empleado, con una longitud fija de 9 caracteres, obligatorio.

apellido1 = fields.Char(string='Primer apellido') # Primer apellido del empleado.

apellido2 = fields.Char(string='Segundo apellido') # Segundo apellido del empleado.

fecha\_nacimiento = fields.Date(string='Fecha de nacimiento') # Fecha de nacimiento del empleado.

fecha\_incorporacion = fields.Date(string='Fecha incorporación', default=lambda self: fields.Datetime.now(), readonly=True) # Fecha de incorporación del empleado, con valor predeterminado de la fecha y hora actual.

foto = fields.Image(string='Foto', max\_width=200, max\_height=200) # Foto del empleado, con límites de tamaño.

# Relación con proyectos y equipos

proyecto\_id = fields.Many2many('creativeminds.proyecto', string='Proyectos') # Relación de muchos a muchos con los proyectos en los que está involucrado el empleado.

departamento = fields.Char(string='Departamento') # Departamento en el que trabaja el empleado.

puesto = fields.Char(string='Puesto') # Puesto o cargo del empleado en la empresa.

equipo\_id = fields.Many2many('creativeminds.equipo', string='Equipos') # Relación de muchos a muchos con los equipos en los que está asignado el empleado.

# Estado de disponibilidad

disponibilidad = fields.Selection([ # Campo para gestionar la disponibilidad del empleado.

('disponible', 'Disponible'), # El empleado está disponible para trabajar.

('asignado', 'Asignado'), # El empleado está asignado a un proyecto.

('parcial', 'Parcialmente Disponible'), # El empleado está parcialmente disponible.

('no\_disponible', 'No Disponible') # El empleado no está disponible.

], string='Disponibilidad', default='disponible') # Valor por defecto es "disponible".

# Restricción en el campo DNI: formato válido

@api.constrains('dni')

def \_check\_dni(self):

regex = re.compile('[0-9]{8}[A-Z]\Z', re.I) # Expresión regular para verificar el formato correcto del DNI (8 dígitos seguidos de una letra).

for record in self:

if not regex.match(record.dni): # Si el formato no es válido, lanza una excepción.

raise ValidationError('ERROR. Formato DNI incorrecto.')

# Restricción SQL: asegura que el DNI sea único en la base de datos.

\_sql\_constraints = [

('DNI\_unico', 'UNIQUE(dni)', "El DNI debe ser único") # Restricción de unicidad en el campo DNI.

]

class Equipo(models.Model):

\_name = 'creativeminds.equipo'

\_description = 'Equipos de Trabajo'

# Campos básicos de información del equipo

equipo\_id = fields.Integer(string='ID Grupo', required=True) # ID único del equipo, obligatorio.

nombre = fields.Char(string='Nombre', required=True) # Nombre del equipo, obligatorio.

empleado\_id = fields.Many2many('creativeminds.empleado', string='Empleado') # Relación de muchos a muchos con los empleados del equipo.

responsable\_id = fields.Many2one('creativeminds.empleado', string='Responsable') # Relación con un solo responsable del equipo.

descripcion = fields.Text(string='Descripcion del equipo') # Descripción del equipo, opcional.

# Campo calculado: número de miembros en el equipo

n\_miembros = fields.Integer(string='Número de Miembros', compute='\_compute\_n\_miembros') # Número de miembros calculado dinámicamente.

# Método para calcular el número de miembros en el equipo

def \_compute\_n\_miembros(self):

for equipo in self:

equipo.n\_miembros = len(equipo.empleado\_id) # Asigna el número de miembros en función de la cantidad de empleados asociados al equipo.

class PanelDeControl(models.Model):

\_name = 'creativeminds.control.panel' # Nombre técnico del modelo en Odoo.

\_description = 'Panel de Control' # Descripción del modelo: se usa para gestionar un panel de control de proyectos.

# Campos básicos de información del panel de control

nombre = fields.Char(string='Nombre', required=True) # Nombre del panel de control, obligatorio.

proyectos\_ids = fields.Many2many('creativeminds.proyecto', string='Proyectos en el Panel') # Relación de muchos a muchos con los proyectos asociados a este panel de control.

Views:

<?xml version="1.0" encoding="UTF-8"?>

<odoo>

<!-- Views -->

<!-- Definición de vistas para la gestión de proyectos -->

<record id="view\_creativeminds\_proyecto\_form" model="ir.ui.view">

<field name="name">Nuevo proyecto</field>

<field name="model">creativeminds.proyecto</field>

<field name="arch" type="xml">

<form string="Proyecto">

<sheet>

<!-- Campos básicos del proyecto -->

<group>

<field name="proyecto\_id" />

<field name="nombre" />

<field name="estado"/>

<field name="empleado\_id" />

<field name="costo\_por\_hora" />

<field name="horas\_asignadas" />

<field name="costo\_total" readonly="1" />

<field name="descripcion" />

<field name="cliente" />

</group>

<!-- Más detalles del proyecto -->

<group>

<field name="porcentaje\_progreso" />

<field name="fecha\_inicio" />

<field name="fecha\_fin" />

<field name="prioridad" />

<field name="responsable\_id" />

<field name="presupuesto\_estimado" />

<field name="costo\_total\_recursos" readonly="1" />

</group>

<!-- Datos adicionales sobre tareas, recursos y riesgos -->

<group>

<field name="tareas\_ids"/>

<field name="recursos\_ids"/>

<field name="riesgos" />

<field name="hitos" />

<field name="dependencias" />

<field name="comentarios" />

<field name="recordatorios\_automaticos" />

</group>

<!-- Botones de acción para ver tareas, miembros, recursos, o duplicar proyecto -->

<group>

<button name="ver\_tareas" string="Ver Tareas" type="object" class="btn-secondary"/>

<button name="ver\_miembros" string="Ver Miembros" type="object" class="btn-secondary"/>

<button name="ver\_recursos" string="Ver Recursos" type="object" class="btn-secondary"/>

<button name="duplicar\_proyecto" string="Duplicar Proyecto" type="object" class="btn-primary" context="{'proyecto\_id': active\_id}" />

</group>

</sheet>

<!-- Funcionalidad de conversación para seguimiento de mensajes -->

<div class="oe\_chatter">

<field name="message\_follower\_ids" widget="mail\_followers" />

<field name="message\_ids" widget="mail\_thread" />

</div>

</form>

</field>

</record>

<!-- Vista tipo árbol para listar proyectos -->

<record id="view\_creativeminds\_proyecto\_tree" model="ir.ui.view">

<field name="name">Lista de proyectos</field>

<field name="model">creativeminds.proyecto</field>

<field name="arch" type="xml">

<tree string="Proyectos">

<field name="proyecto\_id" />

<field name="nombre" />

<field name="estado" />

<field name="fecha\_inicio" />

<field name="fecha\_fin" />

<field name="prioridad" />

<field name="costo\_total" />

<field name="costo\_total\_recursos" />

<field name="responsable\_id" />

<button name="ver\_tareas" string="Ver Tareas" type="object" />

<button name="ver\_miembros" string="Ver Miembros" type="object" />

<button name="ver\_recursos" string="Ver Recursos" type="object" />

</tree>

</field>

</record>

<!-- Vista tipo Kanban para proyectos -->

<record id="view\_creativeminds\_proyecto\_kanban" model="ir.ui.view">

<field name="name">Kanban proyectos</field>

<field name="model">creativeminds.proyecto</field>

<field name="arch" type="xml">

<kanban js\_class="project\_kanban" quick\_create="1" create="1" edit="1" delete="1">

<field name="id" />

<field name="nombre" />

<field name="estado" />

<field name="fecha\_inicio" />

<field name="fecha\_fin" />

<field name="prioridad" />

<field name="costo\_total" />

<field name="costo\_total\_recursos" />

<templates>

<t t-name="kanban-box">

<div class="o\_kanban\_card">

<strong><field name="nombre" /></strong>

<p><small>Estado: <field name="estado" /></small></p>

<p><small>Fecha de Inicio: <field name="fecha\_inicio" /></small></p>

<p><small>Fecha de Fin: <field name="fecha\_fin" /></small></p>

<p><small>Prioridad: <field name="prioridad" /></small></p>

<p><small>Costo Total: <field name="costo\_total" /></small></p>

<p><small>Costo Total de Recursos: <field name="costo\_total\_recursos" /></small></p>

<p><button name="ver\_tareas" string="Ver Tareas" type="object" class="btn-secondary" context="{'proyecto\_id': id}" />

<button name="ver\_miembros" string="Ver Miembros" type="object" class="btn-secondary" context="{'proyecto\_id': id}" /></p>

<div class="o\_kanban\_footer">

<p><button name="ver\_recursos" string="Ver Recursos" type="object" class="btn-secondary" context="{'proyecto\_id': id}" />

<button name="duplicar\_proyecto" string="Duplicar Proyecto" type="object" class="btn-primary" context="{'proyecto\_id': id}" /></p>

</div>

</div>

</t>

</templates>

</kanban>

</field>

</record>

<!-- Vista lista de empleados -->

<record id="view\_creativeminds\_employees\_tree" model="ir.ui.view">

<field name="name">Lista de empleados</field>

<field name="model">creativeminds.empleado</field>

<field name="arch" type="xml">

<tree string="Empleados">

<field name="empleado\_id"/>

<field name="nombre" />

<field name="dni"/>

<field name="apellido1"/>

<field name="apellido2" />

<field name="fecha\_nacimiento" />

<field name="fecha\_incorporacion" />

<field name="puesto" />

<field name="proyecto\_id" />

<field name="departamento" />

<field name="puesto" />

</tree>

</field>

</record>

<!-- Vista formulario de empleados -->

<record model="ir.ui.view" id="view\_creativeminds\_employee\_form">

<field name="name">Nuevo empleado</field>

<field name="model">creativeminds.empleado</field>

<field name="arch" type="xml">

<form >

<sheet string ="Modelo para la gestión de empleadoss">

<div class = "oe\_title">

<h1>Empleado\_<field name="empleado\_id"/></h1>

</div>

<div class = "row">

<div class = "oe\_title mt-4">

<h1>

<field name = "dni" placeholder="12345678Z" class="oe\_inline"/>

</h1>

</div>

<div class ="mt-4">

<field name="foto" widget="image" class="oe\_right" options="{'size':[0,200]}"/>

</div>

</div>

<separator string="Datos personales" colspan="4"/>

<group col="3">

<group >

<field name="apellido1"/>

<field name="fecha\_nacimiento"/>

<field name="fecha\_incorporacion"/>

</group>

<group >

<field name="apellido2"/>

<field name="departamento" />

<field name="puesto" />

</group>

<group >

<field name="nombre" />

<field name="disponibilidad" />

</group>

</group>

<separator string="Equipos" colspan="4"/>

<field name="equipo\_id" widget="many2many\_tags" can\_create="False" can\_write="False">

<tree>

<field name="nombre" />

<field name="responsable\_id" />

<field name="descripcion" />

</tree>

</field>

<separator string="Proyectos" colspan="4"/>

<field name="proyecto\_id" widget="many2many\_tags" can\_create="False" can\_write="False">

<tree>

<field name="nombre" />

<field name="porcentaje\_progreso" />

<field name="fecha\_inicio" />

<field name="fecha\_fin" />

<field name="prioridad" />

<field name="responsable\_id" />

<field name="estado" />

</tree>

</field>

</sheet>

<div class="oe\_chatter">

<field name="message\_ids" widget="mail\_thread"/>

</div>

</form>

</field>

</record>

<!-- Vistas y acciones para equipos, tareas y recursos -->

<!-- Estas vistas permiten gestionar los equipos, tareas y recursos de los proyectos -->

<!-- Vista de equipos -->

<record id="view\_creativeminds\_teams\_tree" model="ir.ui.view">

<field name="name">Lista de equipos</field>

<field name="model">creativeminds.equipo</field>

<field name="arch" type="xml">

<tree string="Equipos">

<field name="nombre" />

<field name="empleado\_id"/>

<field name="responsable\_id" />

<field name="descripcion" />

<field name="n\_miembros" />

</tree>

</field>

</record>

<!-- Vista de tareas -->

<record id="view\_creativeminds\_task\_tree" model="ir.ui.view">

<field name="name">Lista de tareas</field>

<field name="model">creativeminds.tarea</field>

<field name="arch" type="xml">

<tree string="Tareas">

<field name="proyecto\_id" />

<field name="nombre" />

<field name="descripcion" />

<field name="responsable\_id" />

<field name="fecha\_inicio" />

<field name="fecha\_fin" />

<field name="estado" />

</tree>

</field>

</record>

<!-- Vista de recursos -->

<record id="view\_creativeminds\_resource\_tree" model="ir.ui.view">

<field name="name">Lista de recursos</field>

<field name="model">creativeminds.recurso</field>

<field name="arch" type="xml">

<tree string="Recursos">

<field name="nombre" />

<field name="empleado\_id" />

<field name="proyecto\_id" />

<field name="costo\_por\_hora" />

<field name="horas\_asignadas" />

<field name="costo\_total" />

<field name="fecha\_inicio" />

<field name="fecha\_fin" />

<field name="estado" />

</tree>

</field>

</record>

<!-- Vista de panel de control -->

<record id="view\_creativeminds\_control\_panel\_tree" model="ir.ui.view">

<field name="name">Panel de Control</field>

<field name="model">creativeminds.control.panel</field>

<field name="arch" type="xml">

<tree>

<field name="nombre" />

<field name="proyectos\_ids" />

</tree>

</field>

</record>

<!-- Definición de acciones para mostrar las vistas en la interfaz de usuario -->

<!-- Estas acciones permiten abrir las vistas de proyectos, empleados, tareas, recursos, ect -->

<record id="action\_creativeminds\_proyecto" model="ir.actions.act\_window">

<field name="name">Proyectos</field>

<field name="res\_model">creativeminds.proyecto</field>

<field name="view\_mode">tree,form,kanban</field>

</record>

<record id="action\_creativeminds\_employees" model="ir.actions.act\_window">

<field name="name">Empleados</field>

<field name="res\_model">creativeminds.empleado</field>

<field name="view\_mode">tree,form</field>

</record>

<record id="action\_creativeminds\_teams" model="ir.actions.act\_window">

<field name="name">Equipos</field>

<field name="res\_model">creativeminds.equipo</field>

<field name="view\_mode">tree,form</field>

</record>

<record id="action\_creativeminds\_control\_panel" model="ir.actions.act\_window">

<field name="name">Panel de Control</field>

<field name="res\_model">creativeminds.control.panel</field>

<field name="view\_mode">tree,form</field>

</record>

<record id="action\_creativeminds\_task" model="ir.actions.act\_window">

<field name="name">Tareas de Proyecto</field>

<field name="res\_model">creativeminds.tarea</field>

<field name="view\_mode">tree,form</field>

<field name="domain">[('proyecto\_id', '=', active\_id)]</field>

</record>

<record id="action\_creativeminds\_task2" model="ir.actions.act\_window">

<field name="name">Tareas de Proyecto</field>

<field name="res\_model">creativeminds.tarea</field>

<field name="view\_mode">tree,form</field>

</record>

<record id="action\_creativeminds\_resource" model="ir.actions.act\_window">

<field name="name">Recursos</field>

<field name="res\_model">creativeminds.recurso</field>

<field name="view\_mode">tree,form</field>

<field name="domain">[('proyecto\_id', '=', active\_id)]</field>

</record>

<record id="action\_creativeminds\_employee" model="ir.actions.act\_window">

<field name="name">Empleados del Proyecto</field>

<field name="res\_model">creativeminds.empleado</field>

<field name="view\_mode">tree,form</field>

<field name="domain">[('id', 'in', parent.proyecto\_id)]</field>

</record>

<record id="action\_creativeminds\_team\_project" model="ir.actions.act\_window">

<field name="name">Equipos del Proyecto</field>

<field name="res\_model">creativeminds.equipo</field>

<field name="view\_mode">tree,form</field>

<field name="domain">[('id', 'in', parent.proyecto\_id)]</field>

</record>

<!-- Menús para acceder a las distintas secciones del módulo -->

<menuitem

id="menu\_creativeminds\_proyecto\_root"

name="CreativeMinds"

web\_icon="creativeminds\_proyecto,static/description/icon.png"

sequence="10"/>

<!-- Submenús para proyectos, tareas -->

<menuitem

id="menu\_projects"

name="Proyectos"

parent="menu\_creativeminds\_proyecto\_root"

action="action\_creativeminds\_proyecto"

sequence="10"/>

<menuitem

id="menu\_projects\_projects"

name="Ver proyectos"

parent="menu\_projects"

action="action\_creativeminds\_proyecto"

sequence="12"/>

<!-- Menús para tareas, empleados, equipos y panel de control -->

<menuitem

id="menu\_project\_tasks"

name="Ver Tareas"

parent="menu\_projects"

action="action\_creativeminds\_task2"

sequence="15"/>

<menuitem

id="menu\_employees"

name="Empleados"

parent="menu\_creativeminds\_proyecto\_root"

action="action\_creativeminds\_employees"

sequence="20"/>

<menuitem

id="menu\_teams"

name="Equipos"

parent="menu\_creativeminds\_proyecto\_root"

action="action\_creativeminds\_teams"

sequence="30"/>

<menuitem

id="menu\_control\_panel"

name="Panel de Control"

parent="menu\_creativeminds\_proyecto\_root"

action="action\_creativeminds\_control\_panel"

sequence="40"/>

</odoo>