



PROYECTO SGE

CFGS Desarrollo de Aplicaciones
Multiplataforma
Informática y Comunicaciones

Desarrollo del módulo “manage” con Odoo ERP

Año: 2024

Fecha de presentación: 2025

Nombre y Apellidos: Alejandro de Gregorio Miguel
Email: Alejandro.gremig@educa.jcyl.es

Índice

1	Introducción	3
2	Estado del arte	4
3	Descripción general del proyecto	5
4	Diseño de la aplicación	6
5	Pruebas de funcionamiento	18
6	Conclusiones y posibles ampliaciones	21
7	Bibliografía	22

1 Introducción

Un sistema ERP es una herramienta de software que ayuda a organizar y manejar las actividades clave de una empresa. Su principal meta es juntar información y tareas en un solo lugar para hacer todo más eficiente, ahorrar dinero y ayudar a tomar decisiones rápidas basadas en datos.

Características principales:

Integración: conectando diferentes áreas de la empresa.

Automatización: menos trabajo manual y menos errores.

Escalabilidad: ajustarse al crecimiento de tu negocio.

Análisis: Proporciona herramientas avanzadas para la toma de decisiones.

La metodología ágil es una forma de trabajar en proyectos que pone énfasis en ser flexible, colaborativo y entregar constantemente resultados valiosos. Se enfocan en ajustarse rápidamente a los cambios e incluir a los clientes en todas las partes del proceso. SCRUM es uno de los métodos ágiles más populares y organizados.

SCRUM es una estructura adaptable hecha para proyectos complicados y que cambian mucho. Separa el proceso de creación en partes más pequeñas que se llaman sprints, los cuales suelen durar entre 1 y 4 semanas. Cada sprint termina con la salida de un producto o de una mejora funcional.

Funciones principales:

Dueño del producto: identificar y poner en orden de importancia los requerimientos del producto en la lista de tareas.

Scrum Master: ayuda al equipo a seguir las reglas de desarrollo ágil, resuelve problemas y se asegura de que no haya obstáculos en el camino.

Equipo de desarrollo: hacer y entregar nuevos proyectos.

Eventos principales:

Planificación de Sprint: se establecen las metas y actividades del Sprint.

Daily Scrum: Reuniones diarias para coordinar esfuerzos y resolver problemas.

Sprint Review: muestra el trabajo que se ha hecho hasta ahora.

Revisión del Sprint: piensa en lo que salió bien y en lo que se puede mejorar.

SCRUM promueve que todo sea claro, se revise continuamente y se modifique según sea necesario. Es perfecto para proyectos en los que los objetivos pueden cambiar rápidamente.

2 Estado del arte

2.1 ERP

Un ERP es un sistema de software diseñado para integrar y gestionar las actividades más importantes de una empresa en una plataforma centralizada. Los objetivos principales son optimizar los procesos organizacionales, mejorar la toma de decisiones mediante el acceso rápido a datos críticos y reducir costos mediante la automatización.

Con el tiempo, los ERP han evolucionado significativamente. Se originaron en las décadas de 1960 y 1970 como sistemas MRP orientados a la fabricación. Con el tiempo, se expandieron a otras áreas de negocios y finalmente evolucionaron hasta convertirse en los sistemas complejos que conocemos hoy, con versiones en la nube y de código abierto como Odoo que facilitan el acceso y la personalización.

Entre los ERP más populares se encuentran SAP, Oracle ERP Cloud, Microsoft Dynamics y Odoo. Esta última solución se caracteriza por ser modular, accesible y adecuada tanto para grandes como pequeñas empresas. Odoo permite agregar funciones específicas según las necesidades del negocio, lo que la convierte en una solución versátil.

Para este proyecto, hemos optado por desarrollar un módulo en Odoo, utilizando Docker como herramienta principal de instalación. Con Docker, puedes crear fácilmente entornos aislados, garantizando la estabilidad y portabilidad de tus proyectos.

Desde una perspectiva técnica, Odoo utiliza Python como su lenguaje principal y PostgreSQL como su sistema de base de datos. El sistema funciona en una arquitectura cliente-servidor, donde el back-end maneja la lógica de negocios y el front-end proporciona la interfaz de usuario.

La estructura básica de un módulo en Odoo incluye elementos como modelos, definen la estructura de datos; vistas, son responsables de la interfaz; y configuraciones de seguridad que controlan el acceso. Además, el archivo de manifiesto (`__manifest__.py`) sirve como descripción principal del módulo.

2.2 SCRUM

SCRUM es un método de entrega de proyectos ágil ampliamente utilizado, especialmente en entornos con requisitos que cambian con frecuencia. Esta metodología se basa en iteraciones cortas llamadas sprints, que suelen durar entre una y cuatro semanas y tienen como objetivo entregar adiciones funcionales al producto al final de cada sprint.

SCRUM fue desarrollado originalmente en la década de 1990 por Jeff Sutherland y Ken Schwaber y luego se solidificó en 2001 con la publicación del “Manifiesto Ágil”, convirtiéndose en el punto de referencia para proyectos de desarrollo de software y otros campos. Este enfoque es popular debido a su naturaleza colaborativa, flexibilidad para adaptarse al cambio y énfasis en la mejora continua.

El ciclo SCRUM consta de varios eventos clave. La planificación del sprint define los objetivos y las tareas a alcanzar, mientras que las reuniones diarias de Scrum permiten al equipo coordinar actividades y eliminar impedimentos. Al final de cada sprint, habrá una revisión para presentar los resultados, seguida de una retrospectiva para identificar áreas de mejora en el siguiente ciclo.

SCRUM introduce varios conceptos importantes. Las historias de usuario son descripciones breves y centradas en el usuario sobre la funcionalidad requerida. Estas se dividen en tareas específicas que el equipo de desarrollo realiza durante los sprints. En términos de roles, el Product Owner prioriza los requisitos, el Scrum Master garantiza el cumplimiento de las prácticas SCRUM y elimina los impedimentos, y el Equipo de Desarrollo es responsable de construir el producto.

En resumen, SCRUM es un método extremadamente efectivo para proyectos dinámicos, promoviendo la transparencia, la colaboración y la creación continua de valor. Cuando se combina con un sistema ERP como Odoo, puede ser una solución eficaz para una gestión flexible de proyectos.

3 Descripción general del proyecto

3.1 Objetivos

La meta principal de este proyecto es desarrollar un módulo de Odoo que facilite la gestión de los componentes de un proyecto mediante la metodología SCRUM, permitiéndonos agregar, modificar, eliminar y visualizar los distintos componentes de nuestro proyecto. Este módulo contará con campos computados para simplificar el trabajo a aquellos que lo empleen.

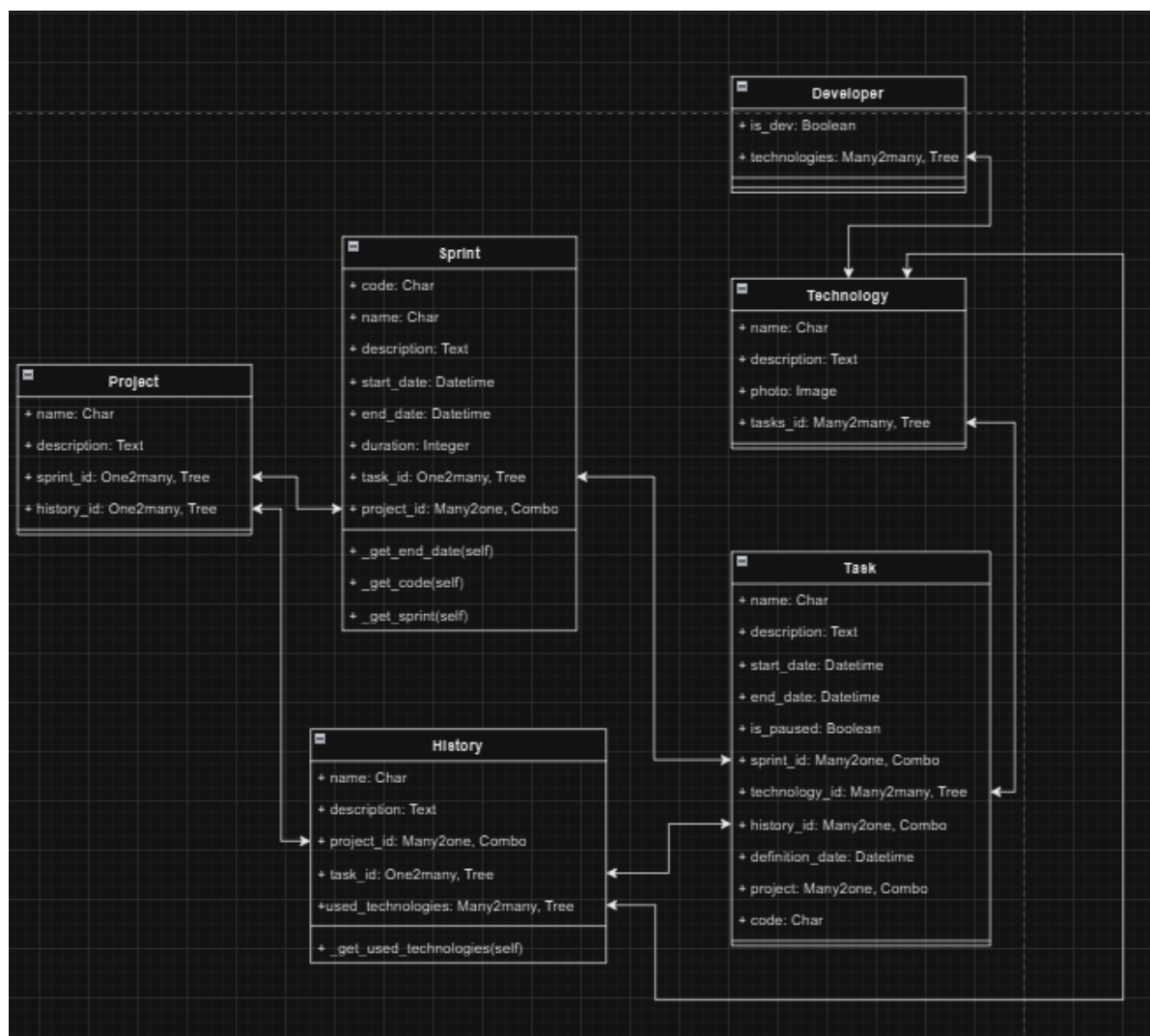
3.2 Entorno de trabajo

Para desarrollar los módulos de Odoo es necesario un IDE cuyo lenguaje de programación sea Python. El IDE que he escogido, en mi caso, es PyCharm, debido a que es un entorno optimizado para Python, con características avanzadas como autocompletado, depuración integrada y herramientas específicas para desarrollo en este lenguaje.

Para desarrollar un módulo de Odoo, debemos tener instalado el propio sistema de Odoo. Para poder facilitar el trabajo tanto en la modificación como para la programación de nuestro módulo hemos utilizado Docker, un sistema operativo que crea unos contenedores que pueden ser ejecutados en local.

4 Diseño de la aplicación

4.1 Modelo relacional de la base de datos



4.2 Partes del proyecto

Este proyecto se compone de varios modelos y vistas:

4.2.1 Clase Project y vista

Este código define el modelo `manage.project` para gestionar proyectos en Odoo, con sus respectivos campos. Se crean vistas XML para mostrar y editar proyectos en formatos de lista y formulario.

```
class project(models.Model):
    _name='manage.project'
    _description='manage.project'

    name = fields.Char(string="Nombre", readonly=False, required=True, help="Introduzca el nombre")
    description = fields.Text(string="Descripción")
    history_id = fields.One2many(string="Histories", comodel_name="manage.history", inverse_name="project_id")
    sprint_id = fields.One2many(string="Sprints", comodel_name="manage.sprint", inverse_name="project_id")
```

```
<odoo>
<data>

<record model="ir.ui.view" id="vista_manage_project_tree">
    <field name="name">vista_manage_project_tree</field>
    <field name="model">manage.project</field>
    <field name="arch" type="xml">
        <tree>
            <field name="name"/>
            <field name="description"/>
        </tree>
    </field>
</record>

<record model="ir.ui.view" id="vista_manage_project_form">
    <field name="name">vista_manage_project_form</field>
    <field name="model">manage.project</field>
    <field name="arch" type="xml">
        <form string="manage_project">
            <sheet>
                <group name="group_top">
                    <field name="name"/>
                    <field name="description"/>
                    <field name="history_id"/>
                    <field name="sprint_id"/>
                </group>
            </sheet>
        </form>
    </field>
</record>

<record model="ir.actions.act_window" id="accion_manage_project_form">
    <field name="name">manage projects</field>
    <field name="type">ir.actions.act_window</field>
    <field name="res_model">manage.project</field>
    <field name="view_mode">tree,form</field>
    <field name="help" type="html">
        <p class="oe_view_nocontent_create">
            Projects
        </p>
        <p> Click <strong> 'Crear' </strong> para añadir nuevos elementos
        </p>
    </field>
</record>

<menuitem name="Projects" id="menu_manage_projects" parent="menu_manage_management" action="accion_manage_project_form"/>
</data>
</odoo>
```

4.2.2 Clase Sprint y vista

Este código define el modelo `manage.sprint` para gestionar sprints en Odoo, con sus respectivos campos y relaciones. Se incluyen métodos que calculan automáticamente el código del sprint y la fecha de fin en función de la duración y la fecha de inicio. Además, se crean vistas XML para mostrar y editar sprints en formatos de lista y formulario.

```
class sprint(models.Model):
    _name='manage.sprint'
    _description='manage.sprint'
    code = fields.Char(string="Codigo", compute="_get_code")
    name = fields.Char(string="Nombre", readonly=False, required=True, help="Introduzca el nombre")
    description = fields.Text(string="Descripción")
    start_date = fields.Datetime(string="Fecha de inicio")
    end_date = fields.Datetime(string="Fecha de fin", compute="_get_end_date", store=True)
    duration = fields.Integer(string="Duracion", default="15")
    task_id = fields.One2many(string="Tasks", comodel_name="manage.task", inverse_name="sprint_id")
    project_id = fields.Many2one("manage.project", string="Project", required=True, ondelete="cascade")

    def _get_code(self):
        for sprint in self:
            if len(sprint.task_id) == 0:
                sprint.code = "FILM_"+str(sprint.id)
            else:
                sprint.code = str(sprint.task_id.name).upper()+"_"+str(sprint.id)

    @api.depends('start_date','duration')
    def _get_end_date(self):
        for sprint in self:
            if isinstance(sprint.start_date,datetime.datetime) and sprint.duration>0:
                sprint.end_date=sprint.start_date+datetime.timedelta(days=sprint.duration)
            else:
                sprint.end_date=sprint.start_date

    @api.depends('code')
    def _get_sprint(self):
        #global sprints, task, found
        for task in self:
            sprints = self.env['manage.sprint'].search([('project.id', '=', task.history.project.id)])
            found = False
            for sprint in sprints:
                if isinstance(sprint.end_date, datetime.datetime) and sprint.end_date > datetime.datetime.now():
                    task.sprint = sprint.id
                    found = True
            if not found:
                task.sprint = False
```



```
<odoo>
<data>

<record model="ir.ui.view" id="vista_manage_sprint_tree">
  <field name="name">vista_manage_sprint_tree</field>
  <field name="model">manage.sprint</field>
  <field name="arch" type="xml">
    <tree>
      <field name="name"/>
      <field name="description"/>
      <field name="start_date"/>
      <field name="end_date"/>
    </tree>
  </field>
</record>

<record model="ir.ui.view" id="vista_manage_sprint_form">
  <field name="name">vista_manage_sprint_form</field>
  <field name="model">manage.sprint</field>
  <field name="arch" type="xml">
    <form string="manage_sprint">
      <sheet>
        <group name="group_top">
          <field name="name"/>
          <field name="description"/>
          <field name="duration"/>
          <field name="start_date"/>
          <field name="end_date"/>
          <field name="task_id"/>
          <field name="project_id"/>
        </group>
      </sheet>
    </form>
  </field>
</record>

<record model="ir.actions.act_window" id="accion_manage_sprint_form">
  <field name="name">manage sprints</field>
  <field name="type">ir.actions.act_window</field>
  <field name="res_model">manage.sprint</field>
  <field name="view_mode">tree,form</field>
  <field name="help" type="html">
    <p class="oe_view_nocontent_create">
      Sprints
    </p>
    <p> Click <strong> 'Crear' </strong> para añadir nuevos elementos
    </p>
  </field>
</record>
<menuitem name="Sprints" id="menu_manage_sprints" parent="menu_manage_management" action="accion_manage_sprint_form"/>
</data>
</odoo>
```

4.2.3 Clase Task y vista

Este código define el modelo manage.task para gestionar tareas en Odoo, con sus respectivos campos y relaciones con otros modelos. Además, se incluyen métodos para calcular automáticamente el código de la tarea y la asignación de sprints. Se crean vistas XML para mostrar y editar tareas en formatos de lista y formulario.

```
class task(models.Model):
    _name='manage.task'
    _description='manage.task'

    name = fields.Char(string="Nombre", readonly=False, required=True, help="Introduzca el nombre")
    description = fields.Text(string="Descripción")
    start_date = fields.Datetime(string="Fecha de inicio")
    end_date = fields.Datetime(string="Fecha de fin")
    is_paused = fields.Boolean(string="¿Pausado?")
    sprint_id = fields.Many2one("manage.sprint", string="Sprint", ondelete="cascade", compute="_get_sprint", store=True)
    history_id = fields.Many2one("manage.history", string="History", required=True, ondelete="cascade")
    technology_id = fields.Many2many(comodel_name="manage.technology",
                                   relation="technologies_tasks",
                                   column1="technologies_ids",
                                   column2="tasks_ids")

    definition_date = fields.Datetime(default=lambda p:datetime.datetime.now())
    project = fields.Many2one("manage.project", related="history_id.project_id", readonly=True)
    code = fields.Char(compute="_get_code")
```

```
<odoo>
<data>

<record model="ir.ui.view" id="vista_manage_task_tree">
    <field name="name">vista_manage_task_tree</field>
    <field name="model">manage.task</field>
    <field name="arch" type="xml">
        <tree>
            <field name="name"/>
            <field name="description"/>
            <field name="start_date"/>
            <field name="end_date"/>
            <field name="is_paused"/>
        </tree>
    </field>
</record>

<record model="ir.ui.view" id="vista_manage_task_form">
    <field name="name">vista_manage_task_form</field>
    <field name="model">manage.task</field>
    <field name="arch" type="xml">
        <form string="manage_task">
            <sheet>
                <group name="group_top">
                    <field name="name"/>
                    <field name="description"/>
                    <field name="start_date"/>
                    <field name="end_date"/>
                    <field name="is_paused"/>
                    <field name="sprint_id"/>
                    <field name="project"/>
                    <field name="history_id"/>
                    <field name="technology_id"/>
                    <field name="definition_date"/>
                </group>
            </sheet>
        </form>
    </field>
</record>

<record model="ir.actions.act_window" id="accion_manage_task_form">
    <field name="name">manage tasks</field>
    <field name="type">ir.actions.act_window</field>
    <field name="res_model">manage.task</field>
    <field name="view_mode">tree,form</field>
    <field name="help" type="html">
        <p class="oe_view_nocontent_create">
            Tasks
        </p>
        <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos</p>
    </field>
</record>

<menuitem name="Tasks" id="menu_manage_tasks" parent="menu_manage_management" action="accion_manage_task_form"/>
</data>
</odoo>
```

4.2.4 Clase History y vista

Este código define el modelo `manage.history` para gestionar historias dentro de Odoo, sus respectivos campos y relaciones. Además, se calcula automáticamente el campo `used_technologies`, que recopila las tecnologías utilizadas en las tareas asociadas a la historia. Se crean vistas XML para mostrar y editar historias en formatos de lista y formulario.

```
class history(models.Model):
    _name='manage.history'
    _description='manage.history'

    name = fields.Char(string="Nombre", readonly=False, required=True, help="Introduzca el nombre")
    description = fields.Text(string="Descripción")
    task_id = fields.One2many(string="Tasks", comodel_name="manage.task", inverse_name="history_id")
    project_id = fields.Many2one("manage.project", string="Project", required=True, ondelete="cascade")
    used_technologies = fields.Many2many("manage.technology", compute="_get_used_technologies")

    def _get_used_technologies(self):
        for history in self:
            technologies = None # Array para concatenar todas las tecnologías. Inicialmente no tiene valor
            for task in history.task_id: # Para cada una de las tareas de la historia
                if not technologies:
                    technologies = task.technology_id
                else:
                    technologies = technologies + task.technology_id
            history.used_technologies = technologies # Asignar las tecnologías a la historia
```

```
<odoo>
<data>

<record model="ir.ui.view" id="vista_manage_history_tree">
    <field name="name">vista_manage_history_tree</field>
    <field name="model">manage.history</field>
    <field name="arch" type="xml">
        <tree>
            <field name="name"/>
            <field name="description"/>
        </tree>
    </field>
</record>

<record model="ir.ui.view" id="vista_manage_history_form">
    <field name="name">vista_manage_history_form</field>
    <field name="model">manage.history</field>
    <field name="arch" type="xml">
        <form string="manage_history">
            <sheet>
                <group name="group_top">
                    <field name="name"/>
                    <field name="description"/>
                    <field name="task_id"/>
                    <field name="project_id"/>
                    <field name="used_technologies"/>
                </group>
            </sheet>
        </form>
    </field>
</record>

<record model="ir.actions.act_window" id="accion_manage_history_form">
    <field name="name">manage histories</field>
    <field name="type">ir.actions.act_window</field>
    <field name="res_model">manage.history</field>
    <field name="view_mode">tree,form</field>
    <field name="help" type="html">
        <p class="oe_view_nocontent_create">
            Histories
        </p>
        <p> Click <strong> 'Crear' </strong> para añadir nuevos elementos
        </p>
    </field>
</record>

<menuitem name="Histories" id="menu_manage_histories" parent="menu_manage_management" actions="accion_manage_history_form"/>
</data>
</odoo>
```

4.2.5 Clase Technology y vista

Este código define el modelo `manage.technology` para gestionar tecnologías en Odoo, con sus respectivos campos y una relación de muchos a muchos con las tareas. Esto permite asociar múltiples tareas con diversas tecnologías. Se incluyen vistas XML para visualizar y editar las tecnologías tanto en formato de lista como de formulario.

```
class technology(models.Model):
    _name='manage.technology'
    _description='manage.technology'

    name = fields.Char(string="Nombre", readonly=False, required=True, help="Introduzca el nombre")
    description = fields.Text(string="Descripción")
    photo = fields.Image(string="Imagen")
    tasks_id = fields.Many2many(comodel_name="manage.task",
                                relation="technologies_tasks",
                                column1="tasks_ids",
                                column2="technologies_ids")
```

```
<odoo>
<data>

<record model="ir.ui.view" id="vista_manage_technology_tree">
    <field name="name">vista_manage_technology_tree</field>
    <field name="model">manage.technology</field>
    <field name="arch" type="xml">
        <tree>
            <field name="name"/>
            <field name="description"/>
            <field name="photo"/>
        </tree>
    </field>
</record>

<record model="ir.ui.view" id="vista_manage_technology_form">
    <field name="name">vista_manage_technology_form</field>
    <field name="model">manage.technology</field>
    <field name="arch" type="xml">
        <form string="manage_technology">
            <sheet>
                <group name="group_top">
                    <field name="name"/>
                    <field name="description"/>
                    <field name="photo"/>
                    <field name="tasks_id"/>
                </group>
            </sheet>
        </form>
    </field>
</record>

<record model="ir.actions.act_window" id="accion_manage_technology_form">
    <field name="name">manage technologies</field>
    <field name="type">ir.actions.act_window</field>
    <field name="res_model">manage.technology</field>
    <field name="view_mode">tree,form</field>
    <field name="help" type="html">
        <p class="oe_view_nocontent_create">
            Technologies
        </p>
        <p> Click <strong> 'Crear' </strong> para añadir nuevos elementos
        </p>
    </field>
</record>

<menuitem name="Technologies" id="menu_manage_technologies" parent="menu_manage_management" action="accion_manage_technology_form"/>
</data>
</odoo>
```

4.2.6 Clase Developer y vista

Este código amplía el modelo res.partner para incluir funcionalidades relacionadas con desarrolladores y asociarlos con tecnologías mediante una relación de muchos a muchos. Se hereda la vista del formulario de contactos de Odoo y se añade una nueva pestaña llamada Devs, donde se muestran las tecnologías asociadas al desarrollador. Además, se configuran acciones y vistas para gestionar desarrolladores (lista y formulario), accesibles mediante un nuevo elemento de menú llamado Devs en el módulo de gestión.

```
class developer(models.Model):
    _name = "res.partner"
    _inherit = "res.partner"

    is_dev = fields.Boolean()
    technologies = fields.Many2many("manage.technology",
        relation="developer_technologies",
        column1="developer_id",
        column2="technologies_id")
```

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
  <data>
    <record model="ir.ui.view" id="manage.devs_partner_form">
      <field name="name">manage devs form</field>
      <field name="model">res.partner</field>
      <field name="inherit_id" ref="base.view_partner_form"/>
      <field name="mode">primary</field>
      <field name="arch" type="xml">
        <xpath expr="//sheet/notebook/page[@name='internal_notes']" position="after">
          <page name="devs" string="Devs">
            <group>
              <group>
                <field name="technologies"/>
              </group>
            </group>
          </page>
        </xpath>
      </field>
    </record>

    <record model="ir.actions.act_window" id="manage.action_developer_window">
      <field name="name">manage developer window</field>
      <field name="res_model">res.partner</field>
      <field name="view_mode">tree,form</field>
    </record>

    <record model="ir.actions.act_window.view" id="manage.action_view_developer_tree">
      <field name="sequence" eval="1"/>
      <field name="view_mode">tree</field>
      <field name="view_id" ref="base.view_partner_tree"/>
      <field name="act_window_id" ref="manage.action_developer_window"/>
    </record>

    <record model="ir.actions.act_window.view" id="manage.action_view_developer_form">
      <field name="sequence" eval="2"/>
      <field name="view_mode">form</field>
      <field name="view_id" ref="manage.devs_partner_form"/>
      <field name="act_window_id" ref="manage.action_developer_window"/>
    </record>

    <menuitem name="Devs" id="menu_manage_devs" parent="manage.menu_manage_management"
      action="manage.action_developer_window"/>
  </data>
</odoo>
```

4.2.7 Menú padre

Este código crea un menú en Odoo con dos niveles. El menú principal se llama Manage y actúa como la raíz, mientras que un submenú llamado Management se encuentra dentro de él. Esto sirve como base para organizar y acceder a diferentes funcionalidades del módulo desde la interfaz de Odoo, proporcionando una estructura clara y jerárquica para las opciones disponibles.

```
<odoo>
  <data>
    <menuitem name="Manage" id="menu_manage_raiz"/>
    <menuitem name="Management" id="menu_manage_management" parent="menu_manage_raiz"/>
  </data>
</odoo>
```

4.2.8 Archivo de configuración y de manifiesto

En el archivo `ir.model.access.csv` se establecen los permisos necesarios para que los usuarios del grupo básico puedan interactuar con los diferentes modelos del módulo. Esto significa que los usuarios tienen permisos completos (lectura, escritura, creación y eliminación) sobre tareas, proyectos, sprints, historias y tecnologías.

El archivo `__manifest__.py` describe las características y dependencias del módulo. Aquí se indica información como su nombre, una descripción general y el propósito del módulo. También se especifica que depende del módulo base, asegurando que tenga las bases necesarias para funcionar. Además, lista los diferentes archivos que se cargan al instalar el módulo, como vistas y plantillas, junto con un apartado para archivos demo en caso de que se necesiten ejemplos de uso.

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_manage_manage,manage.manage,model_manage_manage,base.group_user,1,1,1,1
access_manage_task,manage.task,model_manage_task,base.group_user,1,1,1,1
access_manage_sprint,manage.sprint,model_manage_sprint,base.group_user,1,1,1,1
access_manage_project,manage.project,model_manage_project,base.group_user,1,1,1,1
access_manage_history,manage.history,model_manage_history,base.group_user,1,1,1,1
access_manage_technology,manage.technology,model_manage_technology,base.group_user,1,1,1,1
```

```
# -*- coding: utf-8 -*-
{
    'name': "manage",

    'summary': """
        Short (1 phrase/line) summary of the module's purpose, used as
        subtitle on modules listing or apps.openerp.com""",

    'description': """
        Long description of module's purpose
        """,

    'author': "My Company",
    'website': "https://www.yourcompany.com",

    # Categories can be used to filter modules in modules listing
    # Check https://github.com/odoo/odoo/blob/16.0/odoo/addons/base/data/ir_module_category_data.xml
    # for the full list
    'category': 'Uncategorized',
    'version': '0.1',

    # any module necessary for this one to work correctly
    'depends': ['base'],

    # always loaded
    'data': [
        'security/ir.model.access.csv',
        'views/views.xml',
        'views/models.xml',
        'views/task.xml',
        'views/sprint.xml',
        'views/project.xml',
        'views/history.xml',
        'views/technology.xml',
        'views/developer.xml',
        'views/templates.xml',
    ],
    # only loaded in demonstration mode
    'demo': [
        'demo/demo.xml',
    ],
}
```

4.3 Ampliación del proyecto

En esta ampliación, he añadido la funcionalidad de asignar desarrolladores a las tareas dentro de un sistema de gestión de proyectos en Odoo. A continuación, explicaré detalladamente los cambios realizados y cómo funcionan.

En el modelo task he incorporado un campo nuevo llamado developer_ids, que es un campo de tipo Many2Many. Este campo permite seleccionar varios desarrolladores para cada tarea.

Nueva clase y vista Task:

```
class task(models.Model):
    _name='manage.task'
    _description='manage.task'

    name = fields.Char(string="Nombre", readonly=False, required=True, help="Introduzca el nombre")
    description = fields.Text(string="Descripción")
    start_date = fields.Datetime(string="Fecha de inicio")
    end_date = fields.Datetime(string="Fecha de fin")
    is_paused = fields.Boolean(string="¿Pausado?")
    sprint_id = fields.Many2one("manage.sprint", string="Sprint", ondelete="cascade", compute="_get_sprint", store=True)
    history_id = fields.Many2one("manage.history", string="History", required=True, ondelete="cascade")
    technology_id = fields.Many2many(comodel_name="manage.technology",
                                   relation="technologies_tasks",
                                   column1="technologies_ids",
                                   column2="tasks_ids")

    definition_date = fields.Datetime(default=lambda p: datetime.datetime.now())
    project = fields.Many2one("manage.project", related="history_id.project_id", readonly=True)
    code = fields.Char(compute="_get_code")
    developer_ids = fields.Many2many(
        'res.partner',
        string='Desarrolladores',
        help="Desarrolladores asignados a la tarea"
    )
)
```

```
<record model="ir.ui.view" id="vista_manage_task_form">
    <field name="name">vista_manage_task_form</field>
    <field name="model">manage.task</field>
    <field name="arch" type="xml">
        <form string="manage_task">
            <sheet>
                <group name="group_top">
                    <field name="name"/>
                    <field name="description"/>
                    <field name="start_date"/>
                    <field name="end_date"/>
                    <field name="is_paused"/>
                    <field name="sprint_id"/>
                    <field name="project"/>
                    <field name="history_id"/>
                    <field name="technology_id"/>
                    <field name="definition_date"/>
                    <field name="developer_ids"/>
                </group>
            </sheet>
            <footer>
                <button name="action_print_task_report"
                    string="Generar PDF"
                    type="object"
                    class="btn-primary"/>
            </footer>
        </form>
    </field>
</record>
```


Además he añadido un sistema de reportes que crea un pdf con toda la información de la tarea. Dentro del form de Task aparecerá un botón imprimir que al darle se descargará un pdf con toda la información de las tareas.

```
<odoo>
<data>
<report
  id="task_report"
  model="manage.task"
  string="Reporte de Tarea"
  report_type="qweb-pdf"
  name="manage.report_task_template"
/>

<template id="report_task_template">
  <t t-call="web.html_container">
    <t t-call="web.external_layout">
      <t t-foreach="docs" t-as="doc">
        <div>
          <h2>Tarea: <span t-field="doc.name"/></h2>
          <p><strong>Descripción:</strong> <span t-field="doc.description"/></p>
          <p><strong>Fecha de Inicio:</strong> <span t-field="doc.start_date"/></p>
          <p><strong>Fecha de Fin:</strong> <span t-field="doc.end_date"/></p>
          <p><strong>¿Pausada?:</strong> <span t-field="doc.is_paused"/></p>
          <p><strong>Historia:</strong> <span t-field="doc.history_id.name"/></p>
          <p><strong>Proyecto:</strong> <span t-field="doc.project.name"/></p>
          <p><strong>Tecnologías utilizadas:</strong></p>
          <ul>
            <t t-foreach="doc.technology_id" t-as="tech">
              <li><span t-field="tech.name"/></li>
            </t>
          </ul>
          <p><strong>Desarrolladores asignados:</strong></p>
          <ul>
            <t t-foreach="doc.developer_ids" t-as="dev">
              <li><span t-field="dev.name"/></li>
            </t>
          </ul>
        </div>
      </t>
    </t>
  </template>
</data>
</odoo>
```

Imprimir

Acción

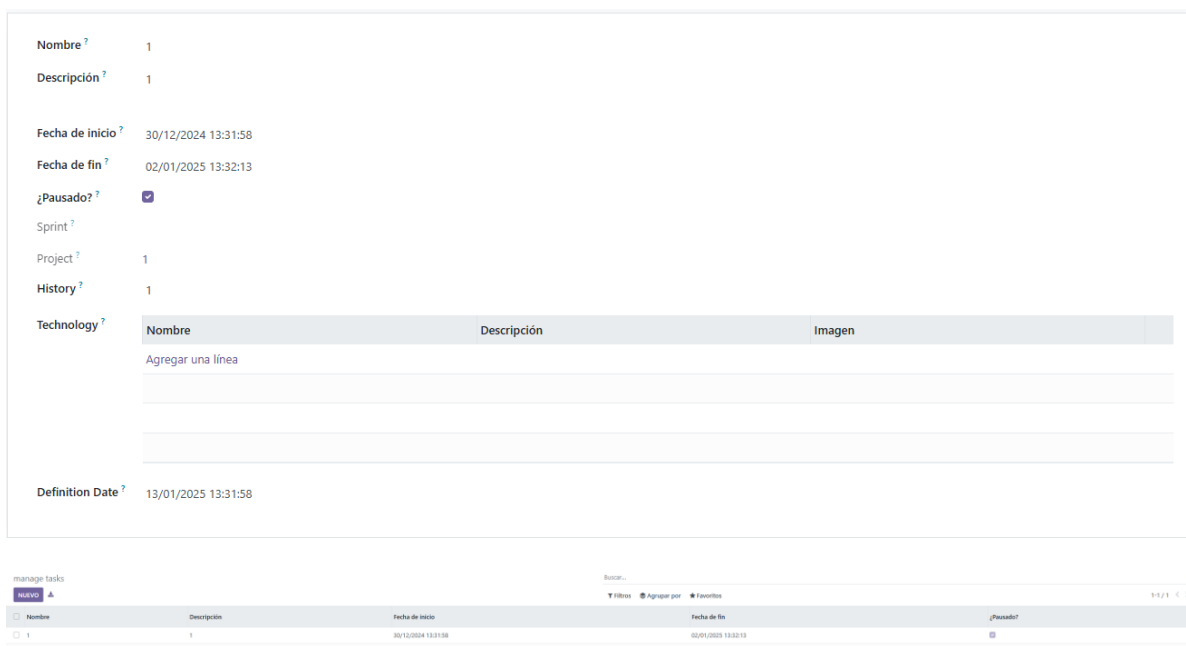
1 / 1 < >

Nuevo

5 Pruebas de funcionamiento

5.1 Tasks

Esto es la vista form del modelo Task, aquí se crea una nueva tarea rellenando los distintos campos que se encuentran aquí. Hay algunos campos que sí o sí tienen que estar rellenos ya que sino no te deja crear la tarea. Los campos de Sprint y Project no se pueden completar directamente desde este formulario sino que desde el propio formulario de cada módulo se tiene que relacionar con una tarea para que se rellene. Después de todo esto se verá todo en la vista Tree.



Nombre ? 1

Descripción ? 1

Fecha de inicio ? 30/12/2024 13:31:58

Fecha de fin ? 02/01/2025 13:32:13

¿Pausado? ? ☒

Sprint ?

Project ? 1

History ? 1

Technology ?

Nombre	Descripción	Imagen
Agregar una línea		

Definition Date ? 13/01/2025 13:31:58

manage tasks

Buscar...

Filtros Agrupar por Favoritos

1/2/1 1 2

Nombre	Descripción	Fecha de inicio	Fecha de fin	¿Pausado?
1	1	30/12/2024 13:31:58	02/01/2025 13:32:13	<input checked="" type="checkbox"/>

5.2 Sprints

Esto es la vista Form del modelo Sprint, al igual que el modelo anterior existen campos que tienen que ser rellenos obligatoriamente para crear el Sprint. Lo que cambia en este modelo son los campos de duración y de fecha de fin. El campo de duración ya viene predefinido a 15 minutos y el campo de fecha de fin se rellena solo una vez elijas la fecha de inicio. Después de todo esto se verá todo en la vista Tree.

Nombre ? 1

Descripción ? 1

Duración ? 15

Fecha de inicio ? 31/12/2024 13:33:15

Fecha de fin ? 15/01/2025 13:33:15

Tasks ?

Nombre	Descripción	Fecha de inicio	Fecha de fin	¿Pau...
Agregar una línea				

Project ? 1

manage sprints

Buscar...

Filtros Agregar por Favoritos

1-1/1 < >

Nombre	Descripción	Fecha de inicio	Fecha de fin
1	1	31/12/2024 13:33:15	15/01/2025 13:33:15

5.3 Projects

Esto es la vista Form del modelo Project, al igual que los modelos anteriores existen campos que tienen que ser rellenados obligatoriamente para crear el Project. Esta es una vista muy sencilla no tiene mucha complicación. Después de todo esto se verá todo en la vista Tree.

Nombre ? 1

Descripción ?

Histories ?

Nombre	Descripción
1	1
Agregar una línea	

Sprints ?

Nombre	Descripción	Fecha de inicio	Fecha de fin
1	1	31/12/2024 13:33:15	15/01/2025 13:33:15
Agregar una línea			

Nombre	Descripción
1	

5.4 Histories

Esto es la vista Form del modelo History, al igual que los modelos anteriores existen campos que tienen que ser rellenados obligatoriamente para crear el History. El campo de used_technologies se rellena automáticamente si la tarea relacionada con la history tiene una tecnología asociada sino saldrá vacío. Después de todo esto se verá todo en la vista Tree.

Nombre ? 1

Descripción ? 1

Tasks ?

Nombre	Descripción	Fecha de inicio	Fecha de fin	¿Pausado?
1	1	30/12/2024 13:31:58	02/01/2025 13:32:13	<input checked="" type="checkbox"/>

Agregar una línea

Project ? 1

Used Technologies ?

Nombre	Descripción	Imagen

manage histories

Buscar...

1/1

5.5 Technologies

Esto es la vista Form del modelo Technology, al igual que los modelos anteriores existen campos que tienen que ser rellenados obligatoriamente para crear el Technology. Tenemos un campo llamado imagen en el que puedes subir una foto. Después de todo esto se verá todo en la vista Tree.

Nombre ? 1

Descripción ? 1

Imagen ? 4.92 Kb

Tasks ?

Nombre	Descripción	Fecha de inicio	Fecha de fin	¿Pausado?
1	1	30/12/2024 13:31:58	02/01/2025 13:32:13	<input checked="" type="checkbox"/>

Agregar una línea

manage technologies

Buscar...

1/1

5.6 Devs

Esto es la vista del modelo Devs. Aquí coge todos los datos del res.partner y a mayores se ha añadido una nueva página llamada Devs con la que se puede relacionar con una tecnología.

6 Conclusiones y posibles ampliaciones

Desarrollar este proyecto fue una experiencia muy satisfactoria para mí, ya que me permitió aplicar muchos de los conocimientos adquiridos durante esta asignatura. Desde la creación de los modelos en Odoo hasta el análisis de sus interrelaciones, logré una comprensión más profunda de cómo se organizan y administran los datos en aplicaciones empresariales reales.

Además, trabajar en este sistema de gestión me ayudó a entender mejor cómo las empresas planifican y supervisan sus proyectos, tareas y recursos. Este proyecto no solo me permitió perfeccionar mis habilidades técnicas, sino que también me enseñó la relevancia de diseñar soluciones modulares y escalables capaces de adaptarse a diversas necesidades.

Este proyecto representó un desafío que me inspiró a continuar aprendiendo y profundizando en el fascinante ámbito de los sistemas de gestión empresarial.

7 Bibliografía

<https://www.odoo.com>

<https://www.atlassian.com/agile/scrum>

<https://www.docker.com>