



IES RIBERA DE CASTILLA

# PROYECTO SGE

---

CFGS Desarrollo de Aplicaciones  
Multiplataforma  
Informática y Comunicaciones

---

## Memoria Proyecto Manage

*Fecha de presentación: 20-12-2024*

**Nombre y Apellidos: Pablo Villagrán González**  
**Email: [pablo.vilgon@educa.jcyl.es](mailto:pablo.vilgon@educa.jcyl.es)**

## Índice

|   |           |
|---|-----------|
| <b>Introducción .....</b>                         | <b>3</b>  |
| <b>Organización de la memoria.....</b>            | <b>4</b>  |
| <b>Estado del arte.....</b>                       | <b>5</b>  |
| <b>Descripción general del proyecto.....</b>      | <b>15</b> |
| <b>Diseño de la aplicación:.....</b>              | <b>17</b> |
| <b>Pruebas de funcionamiento .....</b>            | <b>24</b> |
| <b>Conclusiones y posibles ampliaciones .....</b> | <b>29</b> |
| <b>Bibliografía .....</b>                         | <b>31</b> |

# Introducción

En un entorno empresarial cada vez más dinámico y competitivo, las organizaciones necesitan herramientas y enfoques que les permitan adaptarse rápidamente a los constantes cambios del mercado. En este contexto, los sistemas ERP y las metodologías ágiles han emergido como soluciones clave para mejorar la eficiencia operativa y fomentar la colaboración en los equipos de trabajo.

Los sistemas ERP (Enterprise Resource Planning) son plataformas integradas diseñadas para centralizar y automatizar la gestión de los recursos y procesos de una empresa. Desde áreas como la contabilidad y las finanzas hasta la logística y las ventas, estas herramientas facilitan la toma de decisiones al proporcionar información en tiempo real y permiten una gestión más eficiente de los datos y las operaciones.

Por otro lado, las metodologías ágiles, como Scrum, ofrecen un marco de trabajo flexible y orientado a la entrega continua de valor. Diseñadas específicamente para gestionar proyectos en entornos de alta incertidumbre, estas metodologías se basan en ciclos iterativos e incrementales que promueven la colaboración, la transparencia y la adaptación constante. Scrum, en particular, se ha consolidado como una de las metodologías ágiles más populares debido a su capacidad para organizar equipos multidisciplinarios y centrarse en la consecución de objetivos claros dentro de plazos cortos.

En este trabajo se explora la relación entre los sistemas ERP y las metodologías ágiles, poniendo énfasis en cómo Scrum puede implementarse como una herramienta estratégica para optimizar el desarrollo, la personalización y la adopción de soluciones ERP en empresas de diferentes tamaños y sectores.

# Organización de la memoria

1. Estado del arte, con sus subapartados correspondientes. Trataremos de informar que son los ERP, su evolución, los principales que están funcionando ahora y el que hemos seleccionado. También un poco los tipos de instalaciones que hay y su desarrollo. Acabaremos contando las especificaciones técnicas que tiene y la composición que conlleva un módulo.
2. Estado del arte, también contaremos que es un scrum, su evolución y funcionamiento.
3. Seguiremos describiendo el proyecto que hemos creado y también el entorno donde hemos trabajado.
4. Trataremos de explicar el diseño de la aplicación, junto con las partes del proyecto y sus posibles ampliaciones.
5. Acabaremos probando el funcionamiento de nuestro módulo de odoo y mirando posibles ampliaciones para un futuro.

# Estado del arte

## 1. ERP

### a. Definición de los ERP

Un **erp** (Enterprise Resource Planning), es un software que se utiliza en la gestión Empresarial. Hacen tareas muy variadas, automatizan infinidad de procesos y lo mejor de todo, es que es posible unificar la administración corporativa. Esto mejora el control del empresario sobre su propio negocio.

### b. Evolución de los ERPs

**Década de 1960:** La historia de los sistemas ERP en los negocios comenzó en la industria manufacturera con la aparición de los MRP (Planificación de Requerimientos de Materiales). Estos sistemas se desarrollaron para gestionar y controlar los inventarios, asegurando niveles óptimos de stock en los almacenes.

**Década de 1970:** Durante esta década, los MRP ganaron popularidad gracias a la influencia de IBM. Estas herramientas permitían planificar la cantidad necesaria de materia prima para la producción de bienes y productos. Sin embargo, su implementación requería el uso de grandes y costosos ordenadores centrales, los cuales, a pesar de su tamaño, aún no ofrecían un alto nivel de potencia de cálculo.

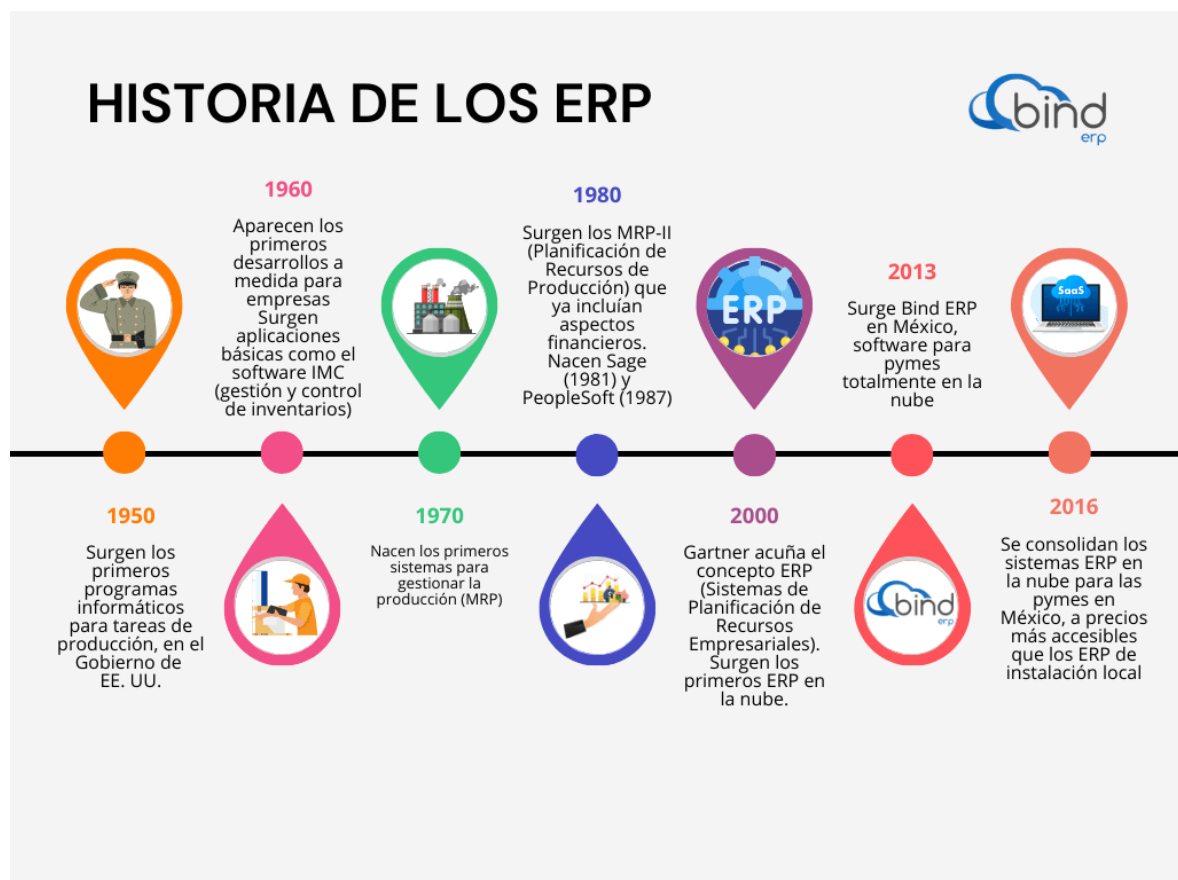
**Década de 1980:** Los sistemas evolucionaron hacia los MRP-II (Planificación de Recursos de Fabricación), integrando funciones más complejas. Estas nuevas versiones no solo gestionaban inventarios, sino que también incorporaban elementos financieros esenciales para la producción, como el costo de la materia prima y la mano de obra.

**Década de 1990:** Surge el ERP tal como lo conocemos hoy en día, destacándose por ofrecer funcionalidades más amplias que las tradicionales centradas en fabricación y finanzas. Estos sistemas estaban conformados por módulos integrados que podían aplicarse a todas las áreas de una empresa.

**Década de 2000:** El desarrollo de internet dio paso a la segunda generación de ERP, la cual permitió la integración con herramientas como los sistemas de gestión de relaciones con clientes (CRM).

**Década de 2010:** Con la llegada del ERP en la nube y la reducción de costos asociados, los sistemas de gestión empresarial comenzaron a ofrecerse bajo el modelo de software como servicio (SaaS). Esto amplió su accesibilidad, haciendo posible su adopción no solo por grandes empresas, sino también por pequeñas y medianas empresas (PyMES).

**Actualidad:** Los sistemas ERP en la nube, como los de Calipso, se han fortalecido gracias a su capacidad de modelización, lo que permite diseñar una arquitectura personalizada para cada organización. Esto facilita soluciones integrales (end-to-end) que abarcan desde las operaciones internas (Back Office) hasta la entrega final de productos o servicios al cliente.



### c. Principales ERP

- 1- Sap Business One: es uno de los más conocidos y ampliamente utilizado en empresas de diferentes tamaños. Ofrece varios módulos para la gestión financiera, compras y ventas. Lo que le hace ideal para adaptarse a las necesidades de tu negocio.
- 2- Oracle NetSuite: es un erp basado en la nube que proporciona una solución muy completa para la gestión empresarial. Su interfaz y sus componentes hacen que sea popular entre las empresas que buscan optimizar sus operaciones y obtener datos en tiempo real.
- 3- Microsoft Dynamics 365: esta combina erp y crm, permitiendo gestionar a las empresas todas sus necesidades desde una plataforma. Es popular entre las pymes por su flexibilidad y escalabilidad, permitiendo crecer las empresas si tener la necesidad de cambiar de sistema.
- 4- Odoo: es una solución erp de código abierto que ofrece una amplia gama de aplicaciones empresariales. Permite empezar con lo básico e ir añadiendo funcionalidades a medida que crecen. Es buena opción, ya que es económica y versátil para pequeñas empresas.
- 5- Sage X3: diseñado para empresas medianas y grandes, con una fuerte orientación hacia la manufacturación y la distribución. Ofrece varias funcionalidades avanzadas, lo que lo convierte en una opción robusta para las empresas con necesidades complejas.

| Programa ERP       | Coste      | Funcionalidades               | Escalabilidad | Soporte en nube |
|--------------------|------------|-------------------------------|---------------|-----------------|
| Sap Business One   | Alto       | Finanzas, ventas, inventarios | Alta          | Sí              |
| Oracle NetSuite    | Alto       | Finanzas, CRM, análisis       | Alta          | Sí              |
| Microsoft Dynamics | Medio-Alto | ERP y CRM integrados          | Alta          | Sí              |
| Odoo               | Bajo-Medio | Modular, contabilidad, ventas | Media-Alta    | Sí              |
| Sage X3            | Medio-Alto | Producción, finanzas          | Alta          | Sí              |

#### **d. ERP seleccionado (Odoo)**

Odoo es un paquete de aplicaciones, en principio dirigido a empresas, tanto grandes, pequeñas o medianas empresas, que permite gestionar de forma global todas las necesidades que se tengan en esa empresa, desde compras, ventas contabilidad, facturación, nóminas, gestión de redes sociales, recursos humanos y una infinidad de aspectos que se pueden gestionar desde el mismo software.

Es una herramienta muy potente, bastante agradable visualmente y muy sencilla de utilizar una vez que hemos aprendido los pasos básicos.

De las muchas características positivas que tiene Odoo, podemos destacar las siguientes:

- Es un software para controlar toda la empresa, por lo que nos evita tener un programa para cada cosa como suele ocurrir, que se tiene un programa para contabilidad, otro para nóminas, otro para llevar la facturación, etcétera. Odoo nos permitirá gestionarlo todo, incluso el correo electrónico y la gestión de la página web, todo lo vamos a poder controlar desde este software ERP.
- Se paga únicamente por módulo utilizado, de tal forma que, si utilizamos unos pocos, no tendremos que pagar por el resto, solo lo haremos por los que realmente vayamos a utilizar.
- Existe una versión local sin coste, la versión Community.
- Permite desarrollar software específico o módulos especiales para nosotros, puesto que es software libre y está programado con Python bajo una un sistema gestor de base de datos PostgreSQL.

#### **Ventajas del módulo Odoo**

- Facilidad de implementación y uso.
- Costos iniciales reducidos en comparación con otros ERP.
- Gran comunidad activa que contribuye a mejoras continuas.
- Flexibilidad para adaptarse a diferentes tipos de industrias y tamaños de empresas.



- e. **Instalación y desarrollo** (*formas de instalación, explicando la que se va a usar para desarrollar el proyecto: Docker*)

Existen varias formas de disponer Odoo para la gestión según las necesidades del desarrollo y producción. Las principales opciones son:

- **Uso de imágenes Docker:** Odoo proporciona imágenes oficiales de Docker que simplifican la configuración y el despliegue del entorno. Permite encapsular la aplicación y sus dependencias en contenedores aislados, siendo ideal para su desarrollo.
- **Uso de plataformas cloud (SaaS):** Odoo también está disponible como servicio alojado en la nube directamente desde Odoo.com. Esto es ideal para empresas que quieren evitar la gestión técnica del sistema, pero no es adecuado para desarrollos avanzados.
- **Instalación desde el código fuente:** Consiste en descargar el código fuente de Odoo desde su repositorio oficial en GitHub. Requiere instalar dependencias como PostgreSQL, Python y librerías específicas. Puede ser compleja de configurar, aunque es flexible para desarrolladores avanzados.
- **Uso de paquetes precompilados:** Odoo ofrece instaladores para diferentes sistemas operativos. Es fácil de usar, pero menos flexible para desarrollos personalizados.

Nosotros hemos optado por utilizar **Docker** para la instalación de Odoo. Para ello:

- Descargamos e instalamos Docker.
- Creamos un archivo docker-compose.yml con la configuración necesaria.
- Generamos las carpetas necesarias en la ruta **C:\**.
- Levantamos los contenedores ejecutando el comando docker-compose up -d en la terminal.
- Accedimos a Odoo a través de la dirección <http://localhost:8069>, desde donde configuramos el entorno para iniciar el desarrollo del proyecto.

## **f. Especificaciones técnicas**

### **i. Arquitectura de Odoo**

Odoo está diseñado bajo una arquitectura modular que permite la personalización y escalabilidad. Su arquitectura se compone de las siguientes capas:

- Base de Datos: Utiliza PostgreSQL como su sistema de gestión de base de datos relacional. Se encarga de almacenar toda la información, incluyendo datos de usuarios, configuraciones y transacciones.
- Servidor (Backend): Desarrollado principalmente en Python, maneja la lógica de negocio y las operaciones en los datos. Proporciona un conjunto de servicios que incluyen autenticación, control de acceso y procesamiento de solicitudes.
- Cliente Web (Frontend): Implementado utilizando tecnologías como JavaScript, HTML y CSS. Permite interactuar con el sistema a través de un navegador web. Utiliza un framework propio llamado QWeb para renderizar vistas.
- Modularidad: Todo el sistema está compuesto por módulos que agregan funcionalidades específicas (gestión de ventas, contabilidad, inventarios, etc.). Estos módulos pueden ser instalados, configurados o desarrollados para adaptarse a necesidades particulares.
- API y Extensibilidad: Proporciona una API que permite la integración con aplicaciones externas y la personalización de módulos existentes.

### **ii. Composición de un módulo**

- Archivo de manifiesto ( manifest .py): archivo principal donde se definen los nombres, descripción, dependencias y otros detalles.
- Modelos (models): contiene las clases de Python que definen la lógica del módulo.
- Vistas (views): define las interfaces de usuario mediante archivos XML. Incluye formularios, listas, menús y acciones relacionadas con los modelos.
- Archivo de seguridad: contiene reglas de acceso y permisos para usuarios y grupos.
- Controladores (controllers): permite manejar interacciones con servicios web o llamadas HTTP.

## 2. SCRUM

### a. Definición de SCRUM

El termino **Scrum**, originalmente surge de un tipo de formación que se realiza en el deporte Rugby, en el cual los jugadores tienen que sacar la pelota sin tocarla con las manos.

Como termino de proyectos, se puede definir como un modelo de desarrollo ágil y flexible, cuya principal prioridad es maximizar el retorno de la inversión (ROI).

Este, está formado por un conjunto de prácticas que nos van a permitir seguir, trabajando en equipo, conseguir agilidad que nos va a favorecer mucho durante el desarrollo y que nos va a permitir obtener mejores resultados.

### b. Evolución

Años 1990: Scrum comenzó a desarrollarse como un marco de trabajo para la gestión de proyectos en entornos complejos. Fue introducido formalmente en 1995 por Ken Schwaber y Jeff Sutherland, quienes presentaron su propuesta en una conferencia de la Asociación de Ingenieros de Software (OOPSLA). Inspirado en métodos de desarrollo iterativos, Scrum surgió como una alternativa flexible y colaborativa frente a los enfoques tradicionales de gestión de proyectos.

Década de 2000: Durante estos años, Scrum ganó popularidad gracias a su inclusión en el Manifiesto Ágil (2001), del cual Schwaber y Sutherland fueron firmantes. Este marco de trabajo comenzó a implementarse en diferentes industrias más allá del software, gracias a su capacidad de adaptarse a diversos contextos empresariales. Además, la creación de roles específicos, como el Scrum Master, el Product Owner y el Equipo de Desarrollo, ayudó a estandarizar su aplicación.

Década de 2010: Con el auge de la transformación digital, Scrum se consolidó como una metodología clave en empresas que buscaban adaptarse rápidamente a los cambios del mercado. En esta etapa, surgieron guías y certificaciones oficiales, como las de Scrum.org y Scrum Alliance, que permitieron la profesionalización de su implementación. Asimismo, se incorporaron conceptos como escalado ágil para aplicar Scrum en proyectos más grandes y equipos distribuidos.

Actualidad: Hoy en día, Scrum es uno de los marcos de trabajo más utilizados en la gestión ágil de proyectos. Su enfoque colaborativo y adaptable sigue evolucionando con la inclusión de prácticas modernas, como DevOps y el uso de herramientas digitales para la gestión remota. Además, ha demostrado ser aplicable no solo en tecnología, sino también en sectores como marketing, educación y salud. Su flexibilidad lo convierte en una herramienta indispensable para equipos que buscan entregar valor continuo en un entorno empresarial dinámico.

### c. Funcionamiento

Scrum en un entorno real:

- Roles Clave del equipo:
  - Product Owner: responsable de definir y priorizar los objetivos del producto, asegurándose de que el equipo trabaje en lo que aporte más valor al cliente.
  - Scrum Master: facilita la implementación del marco scrum, eliminando impedimentos y asegurándose de que el equipo trabaje en lo que más pueda aportar al cliente.
  - Equipo de desarrollo: grupo multifuncional encargado de convertir las ideas en incrementos de producto funcional.

- Definición del trabajo:

El product owner crea y prioriza el Product Backlog, una lista de tareas, funcionalidades o requisitos del proyecto. Los elementos de este se detallan según la necesidad, asegurando que estén listos para ser trabajados en el sprint.

- Planificación del Sprint:

El equipo selecciona los elementos del Backlog que serán trabajados durante el Sprint formando el Sprint Backlog. Los objetivos deben ser claros y alcanzables, el sprint suele durar entre 1 y 4 semanas.

- Ejecución del Sprint:

El equipo trabaja de manera colaborativa en los elementos seleccionados, siguiendo un enfoque iterativo. Cada día se lleva a cabo una reunión breve llamada Daily Scrum (o reunión diaria), en la que los miembros discuten:

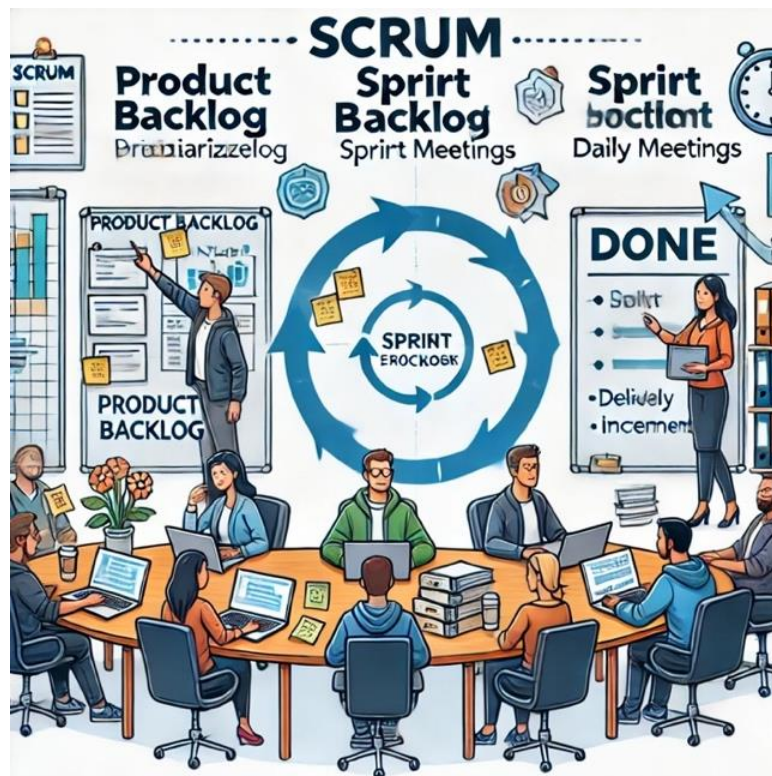
- Qué hicieron el día anterior.
- Qué harán ese día.
- Qué impedimentos enfrentan.

- Entrega del incremento:

Al final del sprint, el equipo entrega un incremento funcional, una versión del producto que puede ser utilizada o evaluada. Esto asegura que el cliente vea avances tangibles de forma constante.

- Revisión y retrospectiva:

- Sprint review: el equipo y los interesados revisan el trabajo realizado, recibiendo retroalimentación para mejorar futuras entregas.
- Sprint retrospective: el equipo analiza su desempeño durante el sprint y define acciones concretas para mejorar la forma de trabajar juntos.



- d. Principales conceptos (*explicar los principales conceptos: proyecto, historias de usuario, sprint, tarea...*)

El módulo **manage** está diseñado para gestionar de manera eficiente proyectos. Los principales conceptos relacionados son:

1. **Proyecto:** Representa un conjunto de actividades planificadas con un objetivo específico. Es la entidad principal que agrupa tareas, historias de usuario y sprints, por ejemplo, el desarrollo de una aplicación o la implementación de un sistema.
2. **Historias de Usuario:** Descripciones simples de las necesidades del usuario. Ejemplo: *"Como gestor, quiero asignar tareas para garantizar que se completen a tiempo."* Ayudan a definir los requisitos del sistema de manera centrada en el usuario.
3. **Sprint:** Intervalos de tiempo (1-4 semanas) en los que se trabaja en un conjunto de tareas específicas. Facilitan la entrega incremental y dividen el proyecto en ciclos manejables.
4. **Tarea:** Unidad básica de trabajo dentro de un proyecto. Cada tarea tiene un responsable, una descripción, un estado y una fecha límite. Ejemplo: *"Implementar autenticación en el sistema."*

# Descripción general del proyecto

- **Objetivos** (breve descripción de lo que se ha pretendido alcanzar con el proyecto);

Al trabajar con Odoo, hemos podido alcanzar nuestros primeros pasos en la implementación de un sistema ERP, lo que nos ha permitido aprender a integrar y automatizar procesos esenciales dentro de una organización. Hemos adquiridos los conocimientos básicos y prácticos. Hemos aprendido a optimizar flujos de trabajo y analizar datos en tiempo real. Además, hemos fortalecido habilidades de resolución de problemas, preparándonos para futuros desafíos en entornos empresarial.

- **Entorno de trabajo** (explicar todas las herramientas utilizadas para desarrollar el proyecto: Docker, navegador, visual studio code...)

Durante el desarrollo del proyecto con Odoo, hemos utilizado varias herramientas importantes que han facilitado tanto la creación como la administración de nuestro espacio de trabajo y el proceso de desarrollo.

A continuación, te ofrezco una explicación más detallada y ampliada sobre las herramientas utilizadas:

- **Docker** ha sido la herramienta principal usada para la creación y administración de nuestros espacios de desarrollo. Esta tecnología de contenedores ha sido clave para asegurar que el espacio de trabajo fuera coherente, ligero y fácil de reproducir en diferentes computadoras. Con Docker, hemos podido configurar contenedores de forma sencilla para ejecutar Odoo y sus requerimientos sin la necesidad de instalar múltiples elementos manualmente, lo cual puede ser un proceso largo y lleno de errores.
- **Visual Studio Code** por otro lado, ha sido nuestro editor de código principal durante el desarrollo de este proyecto. VS Code es un editor ligero pero muy poderoso, perfecto para proyectos de desarrollo que incluyen varios lenguajes y tecnologías, como es el caso de Odoo, que usa Python, XML y otros lenguajes. La interfaz de VS Code es flexible, y su compatibilidad con extensiones ha sido crucial para ajustar el entorno a nuestras necesidades. Hemos utilizado extensiones para facilitar la escritura y depuración de código en Python, así como para trabajar con archivos XML y otros formatos específicos de Odoo. La integración con Git también ha sido útil para la gestión de versiones del código, permitiendo un control efectivo del desarrollo y la colaboración entre los miembros del equipo.

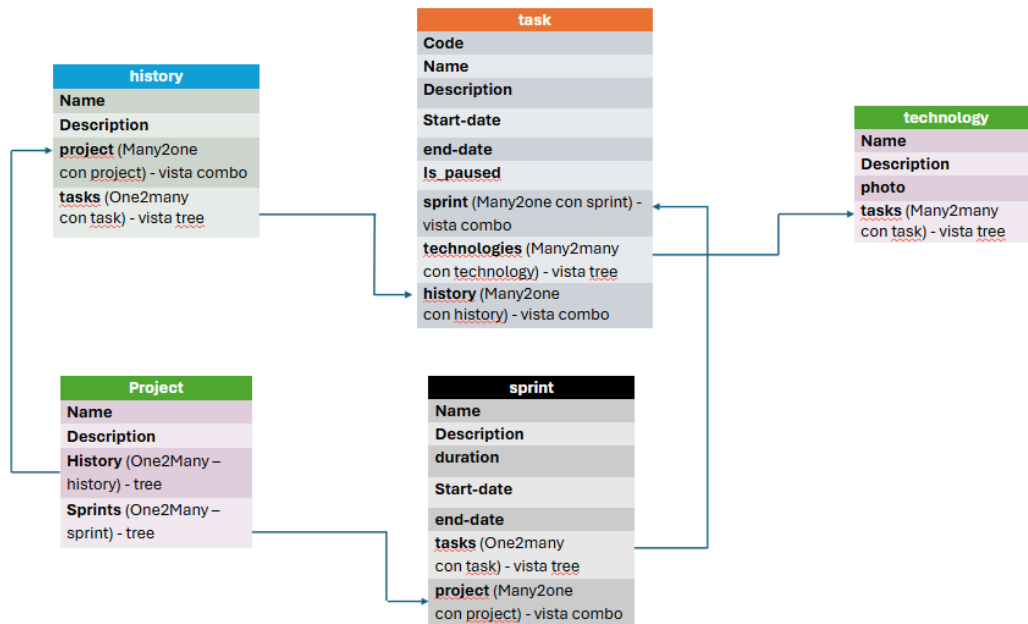
Python, XML y Otras Tecnologías, en lo que respecta a las tecnologías usadas en el desarrollo de módulos de Odoo, Python ha sido el lenguaje principal para la lógica de negocio y la creación de funciones personalizadas. Odoo está basado en Python, lo que nos permitió desarrollar módulos hechos a medida para cumplir los requisitos específicos del proyecto. Además, se utilizó XML para definir las vistas, menús y otros componentes de la interfaz de usuario dentro de Odoo, lo que facilitó la personalización visual de la plataforma. Otras tecnologías y lenguajes también fueron utilizados en menor medida, pero todos ellos ayudaron a la integración efectiva de nuestras soluciones en el ecosistema Odoo.

Conclusión, la mezcla de Docker, Visual Studio Code y el uso de lenguajes como Python y XML nos ha permitido crear un entorno de trabajo muy eficaz, adaptable y fácil de replicar. Esta estructura no solo hizo más sencillo el proceso de desarrollo, sino que también aseguró una mayor eficacia y flexibilidad, lo que resultó en una mejor administración del proyecto y un aprendizaje más rápido sobre Odoo y sus opciones...



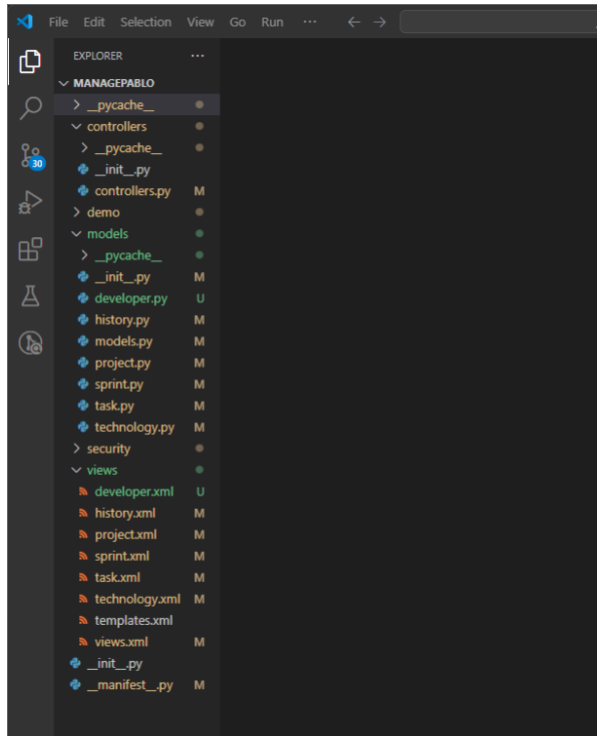
# Diseño de la aplicación:

- Modelo relacional de la BBDD



- **Partes del proyecto** (models, views, security...), explicando brevemente lo que consideréis importante

## Estructura del proyecto manageable



1. Controllers: Contiene los controladores del módulo.
2. Models: Contiene las clases de modelos de datos que extienden o crean nuevas funcionalidades en la base de datos de Odoo.
3. Views: Contiene las vistas XML que definen como se mostraran los datos del modulo en el cliente de Odoo.
4. Security: Incluye un archivo csv que define los permisos de acceso a los modelos para diferentes usuarios.
5. Init.py: Es el archivo que inicializa el módulo, indicando que componentes deben cargarse.
6. Manifest.py: Define la configuración del módulo.

## Models

```
history.py 1 X
models > history.py > ...
1  # -*- coding: utf-8 -*-
2  from odoo import models, fields
3
4  class History(models.Model):
5      _name = 'managepablo.history'
6      _description = 'managepablo.history'
7
8      name = fields.Char(string="Nombre", required=True, help="Introduzca el nombre")
9      description = fields.Text(string="Descripción")
10
11
12      # Muchas historias tienen un proyecto
13      project_id = fields.Many2one("managepablo.project", string="Project", required=True, ondelete="cascade")
14
15      # Una historia tiene muchas tareas
16      task_id = fields.One2many(string="Tasks", comodel_name="managepablo.task", inverse_name="history_id")
17
18      used_technologies = fields.Many2many("managepablo.technology", compute = "_get_used_technologies")
19
20      def _get_used_technologies(self):
21          for history in self:
22              technologies = None
23              for task in history.task_id:
24                  if not technologies:
25                      technologies = task.technologies
26                  else:
27                      technologies = technologies + task.technologies
28              history.used_technologies = technologies
29
```

### 1. Definición del modelo

`_name` define el nombre del modelo

`_description` es una breve descripción del modelo

### 2. Campos de modelo

Name: campo de tipo char para el nombre de la historia. Es obligatorio.

Description: campo de tipo text para una descripción

### 3. Relaciones

Project\_id relación Many2One con el modelo Project. Cada historia esta asociada con un proyecto. El parámetro `ondelete="cascade"` elimina las historias si se elimina el proyecto relacionado.

Task\_id relación One2many con el modelo task. Una historia puede tener muchas tareas.

### 4. Campo calculado

Used\_technologies relación Many2one con el modelo technology. Se calcula automáticamente el método `_get_used_technologies`.

## 5. Método de calculo

El método `_get_used_technologies` recorre todas las tareas asociadas con una historia y agrega sus tecnologías al campo `used_technologies`.

## XML

### 1. Vista de lista

Define una vista de lista (tree) que muestra los campos sus campos. También, tenemos la vista Kanban que consiste en usar tarjetas en un tablero para tener una representación visual de las actividades.

```
<!-- Vista de árbol -->
<record model="ir.ui.view" id="vista_managepablo_history_tree">
  <field name="name">vista_managepablo_history_tree</field>
  <field name="model">managepablo.history</field>
  <field name="arch" type="xml">
    <tree>
      <field name="name"/>
      <field name="description"/>
      <field name="project_id"/>
      <field name="task_id"/>
      <field name="used_technologies"/>
    </tree>
  </field>
</record>
```

### 2. Vista de formulario

Define una vista de formulario (form) con campos de entrada para name, description, Project\_id, task\_id y used\_technologies.

```
<!-- Plantilla formulario tipo form -->

<record model="ir.ui.view" id="vista_managepablo_history_form">
  <field name="name">vista_managepablo_history_form</field>
  <field name="model">managepablo.history</field>
  <field name="arch" type="xml">
    <form string="formulario_sprint" >
      <sheet>
        <group name="group_top">
          <field name="name"/>
          <field name="description"/>
          <field name="project_id"/>
          <field name="task_id"/>
          <field name="used_technologies"/>
        </group>
      </sheet>
    </form>
  </field>
</record>
```

### 3. Acción para listar el modelo

Define la acción para abrir una ventana que muestra el modelo history en modo lista y formulario.

```
<!-- Plantilla action -->

<record model="ir.actions.act_window" id="accion_managepablo_history_form">
  <field name="name">Listado de history</field>
  <field name="type">ir.actions.act_window</field>
  <field name="res_model">managepablo.history</field>
  <field name="view_mode">tree,form</field>
  <field name="help" type="html">
    <p class="oe_view_nocontent_create">
      genero
    </p>
    <p> Click <strong> 'Crear' </strong> para añadir nuevos elementos
    </p>
  </field>
</record>
```

### 4. Menú para acceder a la acción.

Manage de pablo es el menú raíz. Dirección es una categoría dentro del menú raíz. Y history es el elemento de menú que ejecuta la acción **accion\_managepablo\_history\_form** para mostrar las historias.

```
<!-- Top menu item -->
<menuitem name="Manage de Pablo" id="menu_managepablo_raiz"/>

<!-- menu categories -->
<menuitem name="Dirección" id="menu_managepablo_listado" parent="menu_managepablo_raiz"/>

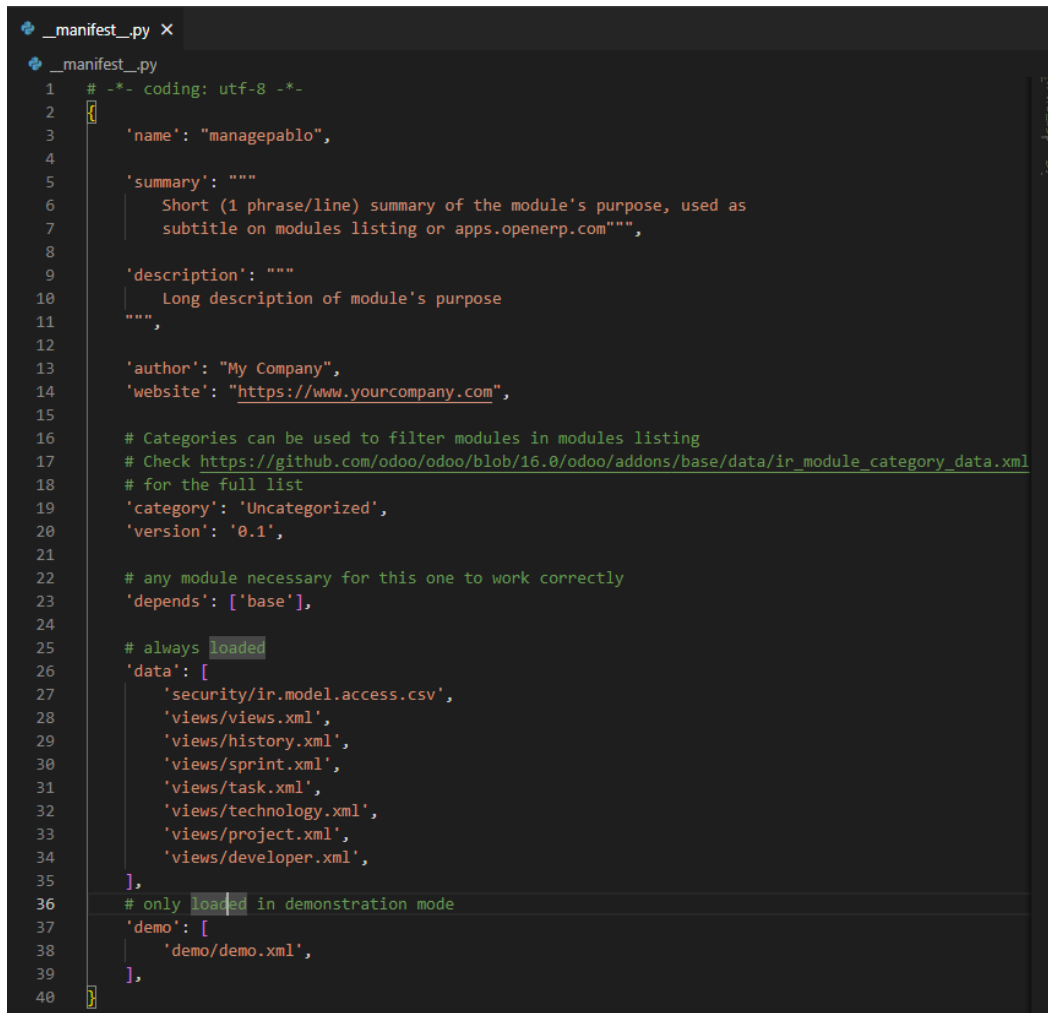
<!-- actions -->
<menuitem name="History" id="menu_managepablo_history" parent="menu_managepablo_listado"
  action="accion_managepablo_history_form"/>
</data>
</odoo>
```

Manifest:

El archivo `__manifest__.py` define la configuración de un módulo en Odoo. Incluye:

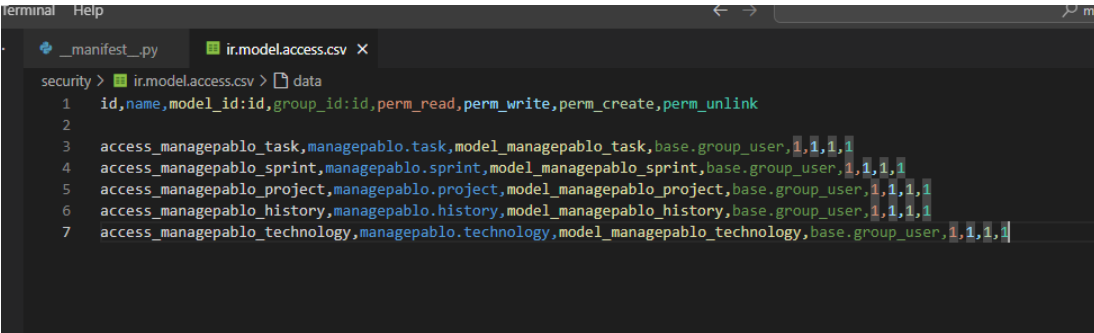
- Información básica: Nombre, resumen, descripción, autor, y versión.
- Categoría: Clasifica el módulo dentro de Odoo.
- Dependencias: Especifica módulos necesarios, como base.
- Datos cargados: Archivos XML para vistas y configuraciones (views.xml, permisos en ir.model.access.csv).
- Datos demo: Archivos opcionales para modo demostración.

Es esencial para que Odoo instale y configure correctamente el módulo.

A screenshot of a code editor showing the content of an `__manifest__.py` file. The file is named `__manifest__.py` and is located in a directory named `managepablo`. The code is written in Python and includes several key sections: a header with encoding, a dictionary for module metadata (name, summary, description, author, website, category, version), a list of dependencies (['base']), a list of data files to be loaded (security/ir.model.access.csv, views/views.xml, views/history.xml, views/sprint.xml, views/task.xml, views/technology.xml, views/project.xml, views/developer.xml), and a list of demo files to be loaded in demonstration mode (demo/demo.xml). The code is formatted with comments and indentation to clearly separate these sections.

```
1 # -*- coding: utf-8 -*-
2
3 'name': "managepablo",
4
5 'summary': ""
6     Short (1 phrase/line) summary of the module's purpose, used as
7     subtitle on modules listing or apps.openerp.com"",
8
9 'description': ""
10     Long description of module's purpose
11     "",
12
13 'author': "My Company",
14 'website': "https://www.yourcompany.com",
15
16 # Categories can be used to filter modules in modules listing
17 # Check https://github.com/odoo/odoo/blob/16.0/odoo/addons/base/data/ir_module_category_data.xml
18 # for the full list
19 'category': 'Uncategorized',
20 'version': '0.1',
21
22 # any module necessary for this one to work correctly
23 'depends': ['base'],
24
25 # always loaded
26 'data': [
27     'security/ir.model.access.csv',
28     'views/views.xml',
29     'views/history.xml',
30     'views/sprint.xml',
31     'views/task.xml',
32     'views/technology.xml',
33     'views/project.xml',
34     'views/developer.xml',
35 ],
36 # only loaded in demonstration mode
37 'demo': [
38     'demo/demo.xml',
39 ],
40
41
```

## Security:



```
terminal Help
ir.model.access.csv X
security > ir.model.access.csv > data
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2
3 access_managepablo_task,managepablo.task,model_managepablo_task,base.group_user,1,1,1,1
4 access_managepablo_sprint,managepablo.sprint,model_managepablo_sprint,base.group_user,1,1,1,1
5 access_managepablo_project,managepablo.project,model_managepablo_project,base.group_user,1,1,1,1
6 access_managepablo_history,managepablo.history,model_managepablo_history,base.group_user,1,1,1,1
7 access_managepablo_technology,managepablo.technology,model_managepablo_technology,base.group_user,1,1,1,1
```

El archivo ir.model.access.csv define permisos para los modelos del módulo. Contiene las siguientes columnas:

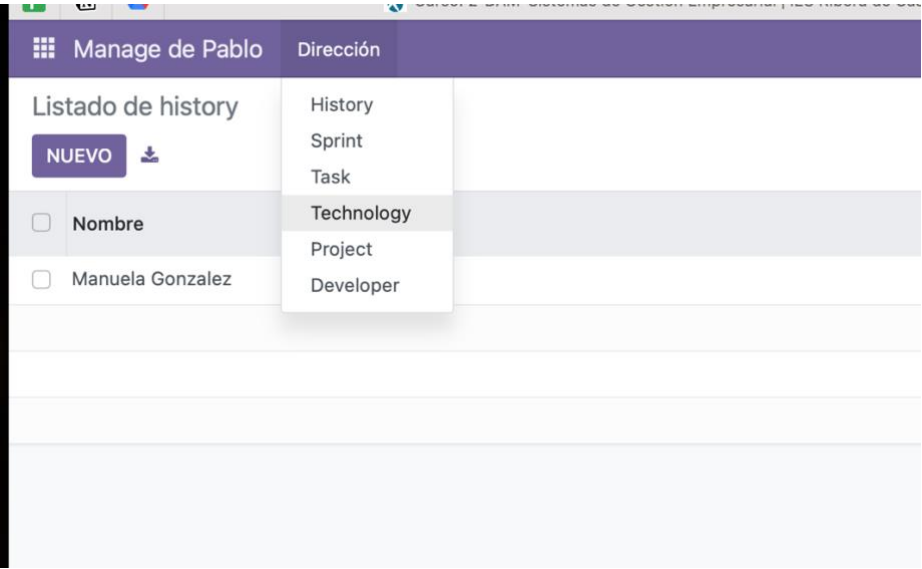
- id: Identificador único del permiso.
  - name: Nombre del permiso.
  - model\_id:id: Modelo al que aplica el permiso (por ejemplo, managepablo.task).
  - group\_id:id: Grupo de usuarios al que se aplica (por ejemplo, base.group\_user).
  - perm\_read: Permiso de lectura (1 = permitido, 0 = denegado).
  - perm\_write: Permiso de escritura.
  - perm\_create: Permiso de creación.
  - perm\_unlink: Permiso de eliminación.
- **Ampliación del proyecto**, explicando detalladamente el objetivo de la ampliación, el desarrollo...

El objetivo es integrar Odoo con otras plataformas y servicios externos mediante APIs. Esto permitirá automatizar el intercambio de datos, mejorando la eficiencia operativa y ofreciendo una experiencia mas completa a los usuarios.

Habría que introducir Postman para probar las APIs y verificar su correcto funcionamiento. Obteniendo reducción de trabajo manual y adaptando el sistema a las necesidades cambiantes del negocio.

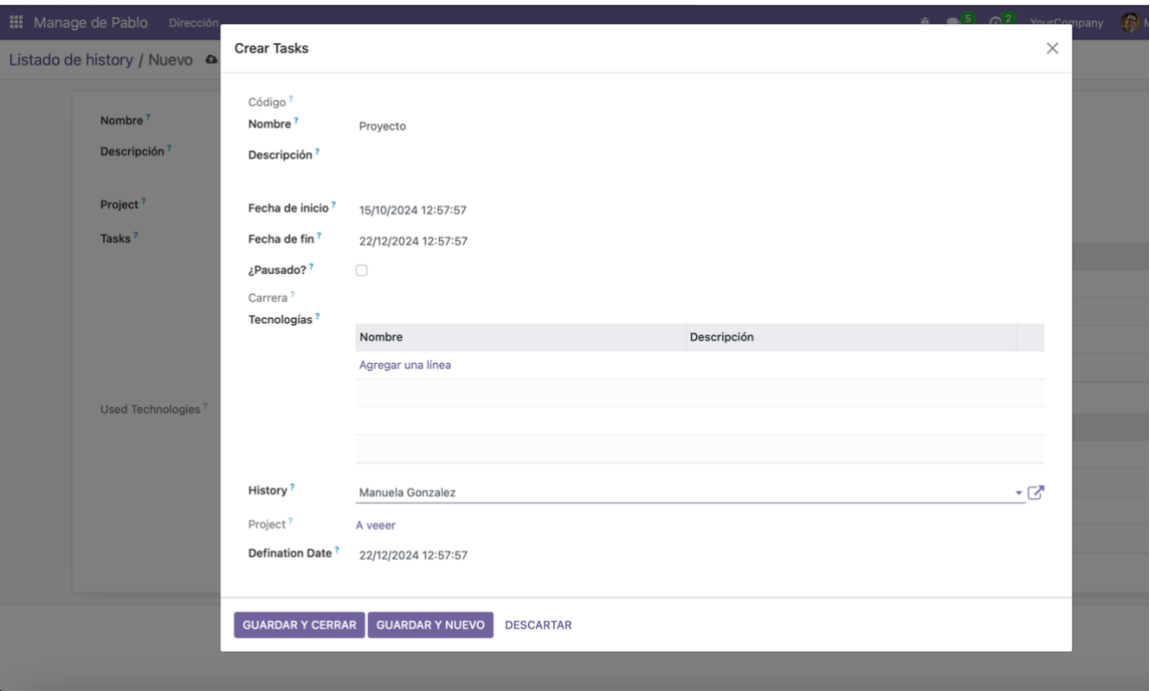
# Pruebas de funcionamiento

Menú principal de gestión, donde muestra el menú de navegación. Cada opción te lleva a una lista o formulario con esa entidad.



Creación de una tarea:

Tenemos varios elementos claves, código y nombre (identificadores únicos), fecha inicio y fecha fin, pausado (si esta activa la tarea o no), history para vincular la tarea a una historia.





Creación de una historia:

Proporciona un resumen detallado de una historia con sus componentes claves para un proyecto. Como vemos, podemos implementar la tecnología que deseemos.

Listado de history / Nuevo

Nombre?Entrega sge

Descripción?Entrega para el domingo de sge

Project?A veeer

Tasks?

| Nombre            | Descripción |
|-------------------|-------------|
| Proyecto          |             |
| Agregar una línea |             |
|                   |             |

Used Technologies?

| Nombre | Descripción |
|--------|-------------|
|        |             |
|        |             |
|        |             |

Creación de un sprint

Listado de sprint / Nuevo

Nombre?Local

Descripción?

Fecha de inicio?11/12/2024 12:58:55

Fecha de fin?21/12/2024 12:58:55

Tareas?

| Nombre            | Descripción |
|-------------------|-------------|
| Agregar una línea |             |
|                   |             |
|                   |             |

Duración (días)?10

Project?A veeer

## Listado technology:

Listado de technology / Nuevo

Nombre ?

Juan

Descripción ?

Imagen ?

PD94bWwgdmlvc2lvdj0lMS4wIlBibmNvZGluZz0lVVRGLTgiPz4KPHByb2plY3QgeG1sbmM9Imh0dHA6Ly9tYXZib5hcGFjaGUub3JnL1BPTS80LjAuMCIKICAgICAglCAgeG1sbmM6eHN

Tareas ?

| Nombre            | Descripción |
|-------------------|-------------|
| Proyecto          |             |
| Agregar una línea |             |
|                   |             |
|                   |             |

Gestión de usuarios:

Muestra la ficha de un usuario o contacto. Mostrando todos sus datos. También, podemos crear uno desde cero.

**Módulo de Ventas**

Nuevo

Acción

---

\$ 0 Ventas
 0,00 Facturado

☒ Individuo ☐ Compañía

p. ej. Brandom Freeman

Nombre de la empresa...

|   |  |
|---|--|
| <p><b>Contacto</b></p> <p>Calle...<br/>Calle 2...<br/>Provincia<br/>País</p> <p><b>NIIF *</b> p. ej., ESA00000000</p> | <p><b>Puesto de trabajo *</b> p. ej. director de ventas</p> <p><b>Teléfono *</b></p> <p><b>Móvil *</b></p> <p><b>Correo electrónico *</b></p> <p><b>Sitio web *</b> p. ej. https://www.odoo.com</p> <p><b>Título *</b> p. ej. Señor</p> <p><b>Etiquetas *</b> p. ej., "B2B", "VIP", "consultoría", ...</p> |
|---|--|

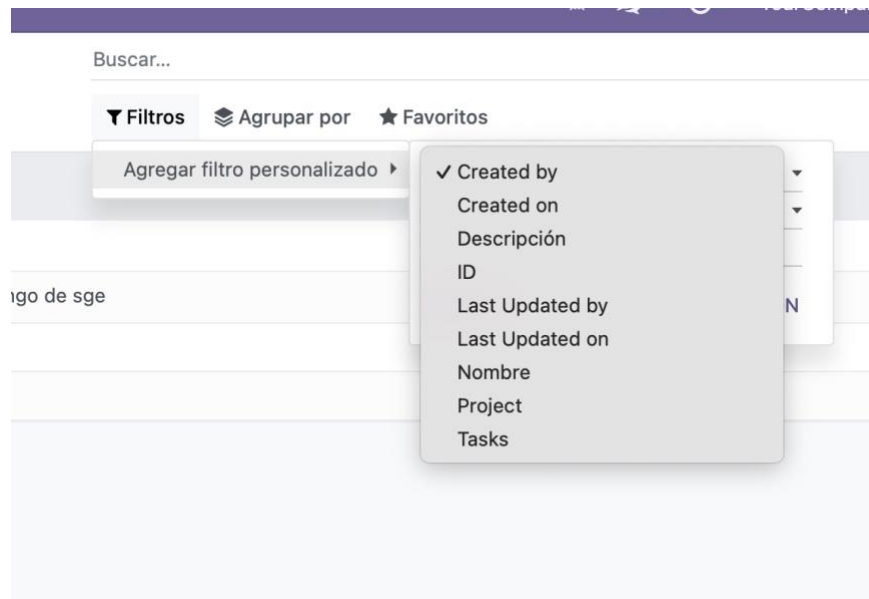
Contatos y direcciones    Ventas y compras    Facturación / Contabilidad    Notas internas    Devs

AGREGAR

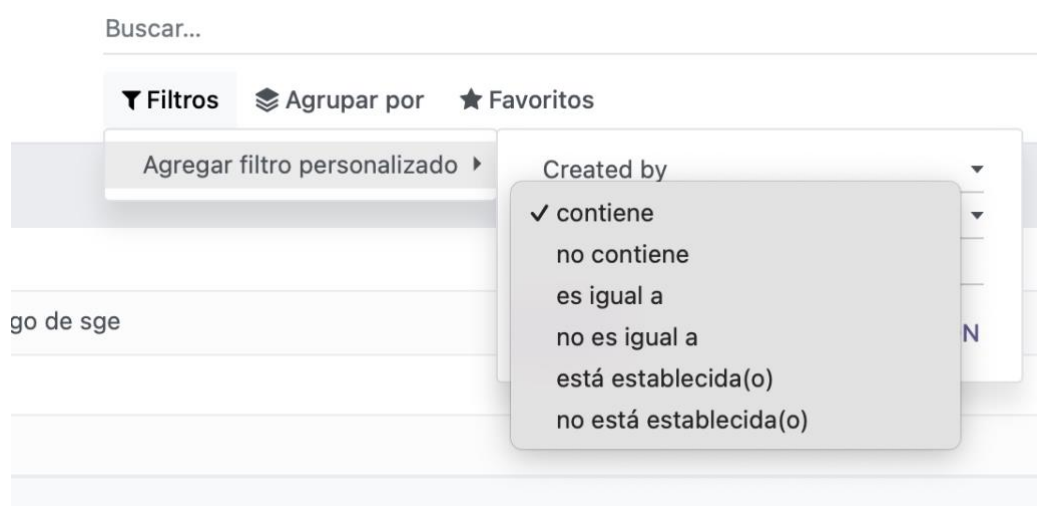
[Enviar mensaje](#)    Registrar una nota    Actividades    Seguir

Filtros:

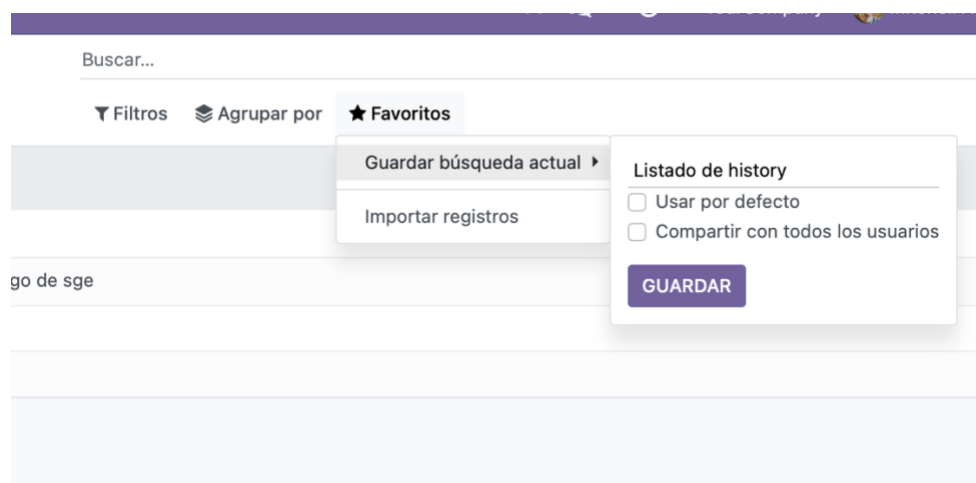
Podemos filtrar por varias opciones, en esta primera imagen por el usuario que lo creo, por la descripción, el id, nombre...



En este filtro, podemos diferenciarlo por registros, como contiene una palabra o frase, es igual a una coincidencia específica...



Observamos el menú de favoritos, que nos permite guardar configuraciones de búsqueda personalizadas. También, pudiendo importar registros externos al sistema.



## Conclusiones y posibles ampliaciones

Hemos logrado integrar procesos clave en una plataforma centralizada, optimizando áreas como inventario, ventas, compras... con este proyecto, hemos podido adquirir habilidades técnicas y prácticas esenciales, preparándonos para abordar desafíos más complejos en el ámbito empresarial y el desarrollo software.

Una de las ampliaciones que implementé en este proyecto fue la funcionalidad para exportar datos en formato JSON. Esta característica es pensada con el objetivo de ofrecer mayor flexibilidad y accesibilidad a los usuarios, permitiéndoles exportar cualquier conjunto de registros que pudieran necesitar para su análisis o integración con otras herramientas. Ya sea que se trate de productos, ventas, clientes o cualquier otra información gestionada dentro de Odoo, esta funcionalidad facilita la obtención de datos estructurados en un formato comúnmente utilizado en la industria para la interoperabilidad entre sistemas.

La implementación de esta funcionalidad no solo mejora la experiencia del usuario, sino que también abre puertas a futuras ampliaciones y mejoras.

Finalmente, considerando la evolución de las tecnologías y la creciente importancia de la integración con otras plataformas, una ampliación futura podría ser la integración de Odoo con sistemas de análisis y visualización de datos, como Power BI o Tableau. De este modo, los datos exportados en JSON podrían ser consumidos directamente por estas herramientas, ofreciendo a los usuarios finales una forma más visual y dinámica de analizar la información y tomar decisiones basadas en datos en tiempo real.

En conclusión, este proyecto ha logrado resultados significativos tanto en términos de optimización de procesos como en la adquisición de habilidades técnicas fundamentales. A través de las ampliaciones implementadas y las que se vislumbran en el futuro, se podría seguir mejorando la plataforma y adaptándola de manera aún más eficiente a las necesidades cambiantes de la empresa, preparando el terreno para soluciones más avanzadas y escalables.

## Código

```
def action_managepablo_history_export_json(self):
    if not self:
        return []
    data_list = []

    for task in self:
        data_list.append({
            'name': task.name,
            'description': task.description,
            'start_date': task.start_date.strftime('%Y-%m-%d %H:%M:%S') if task.start_date else None,
            'end_date': task.end_date.strftime('%Y-%m-%d %H:%M:%S') if task.end_date else None,
            'is_pause': task.is_paused,
            'carrera_id': task.carrera_id.id if task.carrera_id else None,
            'tecnologias_id': [tech.id for tech in task.tecnologias_id],
            'history_id': task.history_id.name,
            'proyect': task.proyect_id.name if task.proyect_id else None,
            'defination_date': task.defination_date.strftime('%Y-%m-%d %H:%M:%S') if task.defination_date else None,
        })

    datos_json = json.dumps(data_list, indent=4)
    archivo_json = base64.b64encode(datos_json.encode('utf-8'))

    archivoAdjunto = self.env['ir.attachment'].create({
        'name': 'tasks_export.json',
        'type': 'binary',
        'datas': archivo_json,
        'res_model': 'managepablo.task',
        'res_id': self.ids[0] if self else None,
        'mimetype': 'application/json'
    })

    return {
        'type': 'ir.actions.act_url',
        'url': f'/web/content/{archivoAdjunto.id}?download=true',
        'target': 'self'
    }
```

```
<record model="ir.ui.view" id="vista_managepablo_task_tree">
  <field name="name">vista_managepablo_task_tree</field>
  <field name="model">managepablo.task</field>
  <field name="arch" type="xml">
    <tree>
      <field name="name"/>
      <field name="description"/>
      <field name="name"/>
      <field name="description"/>
      <field name="start_date"/>
      <field name="end_date"/>
      <field name="is_paused"/>
      <field name="carrera_id"/>
      <field name="history_id"/>
      <button name="action_managepablo_history_export_json" type="object" string="JSON" class="oe_higlight"/>
    </tree>
  </field>
</record>
```

| Manage de Pablo Management |             |          |             |                     |                               |           |         |             |      |
|----------------------------|-------------|----------|-------------|---------------------|-------------------------------|-----------|---------|-------------|------|
| Listado de task            |             |          |             |                     | Buscar...                     |           |         |             |      |
| NUEVO                      |             |          |             |                     | Filtros Agrupar por Favoritos |           |         |             |      |
| 1-1/1                      |             |          |             |                     |                               |           |         |             |      |
| Nombre                     | Descripción | Nombre   | Descripción | Fecha de inicio     | Fecha de fin                  | ¿Pausado? | Carrera | History     |      |
| Proyecto                   |             | Proyecto |             | 15/10/2024 12:57:57 | 22/12/2024 12:57:57           |           |         | Entrega sge | JSON |

```
(44) tasks_export-2
(44) tasks_export-2 > No Selection

1  [
2    {
3      "name": "Proyecto",
4      "description": false,
5      "start_date": "2024-10-15 10:57:57",
6      "end_date": "2024-12-22 11:57:57",
7      "is_pause": true,
8      "carrera_id": null,
9      "tecnologias_id": [
10       1
11     ],
12     "history_id": "Entrega sge",
13     "proyect": "A veeer",
14     "defination_date": "2024-12-22 11:57:57"
15   }
16 ]
17
```

# Bibliografía

<https://openwebinars.net/blog/que-es-odoo/>

<https://www.odoo.com/documentation/16.0/developer.html>

<https://odoo-development.readthedocs.io/en/latest/dev/py/fields.html>

<https://castilloinformatica.es/wiki/index.php?title=Odoo>

## **Criterios calificación memoria proyecto manage:**

### **Presentación formal (20%)**

- Se ajusta a los requerimientos formales establecidos: portada, bibliografía, formato entrega, nombre archivo
- Se incluye un índice estructurado y coherente con el contenido del proyecto
- El texto está bien redactado, no presenta incoherencias gramaticales ni faltas de ortografía
- Presenta el proyecto en forma y plazo establecidos (1 punto menos por cada día de retraso)

### **Contenidos (40%)**

- Originalidad del tema elegido (contenidos originales, no repetidos)
- Grado de dificultad en orden a la investigación de contenidos
- Grado de profundización en la investigación de contenidos
- Explicación clara y concisa, con capturas, explicaciones breves...
- Incluye todos los apartados requeridos en el proyecto

### **Defensa oral (40%)**

- Se ajusta al tiempo marcado
- Objetivo del proyecto
- Puntos esenciales de la resolución
- Conclusiones finales
- Grado de conocimiento y dominio de los contenidos expuestos
- Lenguaje técnico utilizado