



PROYECTO SGE

CFGS Desarrollo de Aplicaciones
Multiplataforma
Informática y Comunicaciones

**“Desarrollo del módulo “manage” con
Odoo ERP; para gestionar proyectos usando
metodologías ágiles: scrum”.**

Año: 2024

Fecha de presentación: 2025

Nombre y Apellidos: Isaac González Adeva
Email: isaac.gonade@educa.jcyl.es

ÍNDICE

1- Introducción	3
2- Organización de la memoria	4
3- ESTADO DEL ARTE	5
4- CONTINUACION PROYECTO MANAGE	8
5- Modelo relacional de la BBDD	10
6- Partes del proyecto	11
7- POSIBLE AMPLIACIÓN	17
8- PRUEBAS DE FUNCIONAMIENTO	18
9- CONCLUSIONES	21
10- BIBLIOGRAFÍA.....	22

1- Introducción

Un ERP es un sistema de software que ayuda a las organizaciones a optimizar sus procesos de negocio centrales con una visión unificada de la actividad. Una manera sencilla de comprender qué es un ERP es compararlo con un cuerpo humano. Así como el cuerpo tiene procesos esenciales para su salud y funcionamiento como el sueño y la digestión, una empresa cuenta con actividades fundamentales, como la cadena de suministro y las ventas. Aunque estos procesos suelen operar de manera independiente, están interconectados y afectan al rendimiento global del cuerpo, en este caso, del negocio. Esto implica que los problemas en un área pueden influir significativamente en las demás.

El software ERP funciona como el sistema nervioso central de la empresa, facilitando la gestión eficiente de estos procesos y operaciones clave al integrarlos en un único sistema conectado.

Las metodologías ágiles significan una nueva mentalidad y forma de hacer las cosas, que nos permite adaptarnos constantemente sin perder el foco en los resultados. Permiten ajustar sobre la marcha, manteniendo a tu equipo enfocado en pequeños objetivos que suman al resultado final. Algunos ejemplos pueden ser Miro que es una herramienta que facilita la colaboración y hace que las ideas fluyan. Otro ejemplo es Cisco que reduce los tiempos de entrega sin sacrificar la calidad de los productos. Extreme Programming XP, Kanban, Agile Inception, Design Sprint, Lean Startup, Crystal, y en la que me voy a centrar SCRUM.

SCRUM ha mejorado la colaboración entre sus equipos y aumentado la satisfacción de sus clientes. Esta metodología, conocida como la "metodología del caos", se basa en un enfoque de desarrollo incremental. Esto significa que cualquier ciclo de desarrollo de un producto o servicio se divide en "microproyectos", que a su vez se organizan en etapas como análisis, desarrollo y pruebas. Durante la fase de desarrollo, se realizan interacciones conocidas como Sprints, que consisten en entregas parciales y regulares del producto final. Scrum es especialmente adecuado para gestionar proyectos complejos que requieren flexibilidad y rapidez en la obtención de resultados. Su estrategia se centra en gestionar y corregir errores que podrían surgir en desarrollos extensos, apoyándose en reuniones frecuentes para garantizar el cumplimiento de los objetivos definidos. Las reuniones son el núcleo de esta metodología y se dividen en: reuniones de planificación, diarias, de revisión y de retrospectiva. Esta última, considerada la más importante, se lleva a cabo al finalizar cada Sprint para reflexionar sobre el progreso y proponer mejoras. Los pilares fundamentales de Scrum son la innovación, la flexibilidad, la competitividad y la productividad.

2- Organización de la memoria

- **Portada**

Incluye el título del proyecto, mi nombre, la fecha de entrega, y mi correo.

- **Índice**

Lista todos los apartados de la memoria con sus páginas.

- **Introducción**

Presentación del proyecto. Doy contexto de los ERP y el SCRUM y la importancia de las herramientas ágiles.

- **Estado del arte**

Explicación de los conceptos básicos. Se incluyen los conceptos de ERP, SCRUM, y otras herramientas relevantes. Se describen de manera detallada los sistemas ERP, su evolución, y cómo la metodología SCRUM se relaciona con la gestión de proyectos.

- **Continuación del proyecto**

Aquí se describe el proceso utilizado para desarrollar el proyecto. Detallando sobre Odoo como ERP, la implementación de SCRUM, y las herramientas como Docker y PyCharm. También se explica cómo se gestionó el proyecto utilizando estas tecnologías.

- **Modelo relacional de la base de datos**

Esta sección muestra una imagen de cómo está estructurada la base del proyecto.

- **Partes del proyecto**

Descripción detallada de los componentes que forman el proyecto. Incluye:

Models: Explicación de los modelos de datos (por ejemplo, Task, Sprint, Project).

Views: Descripción de las vistas que utiliza el usuario.

Security: Explicación de las medidas de seguridad implementadas para gestionar el acceso a los modelos.

Otros componentes como controladores, APIs, y archivos de datos.

- **Ampliaciones**

Ideas sobre las posibles mejoras para que los módulos tengan mejores funciones o para integrar nuevos módulos.

- **Pruebas**

Las pruebas realizadas para asegurar que el sistema funciona, las pruebas de los módulos, las funcionalidades implementadas.

- **Conclusiones**

Reflexión final sobre lo aprendido durante el desarrollo del proyecto y las implicaciones de la implementación de un ERP y SCRUM en la empresa o donde se use.

- **Bibliografía**

Listado de todas las fuentes de información consultadas durante el desarrollo del proyecto.

3- ESTADO DEL ARTE

1. ERP

Definición de los ERP

Los sistemas de Planificación de Recursos Empresariales (ERP, por sus siglas en inglés) son soluciones de software que integran y gestionan los procesos clave de una organización. Estas herramientas centralizan la información de diferentes departamentos, como finanzas, recursos humanos, logística, producción y ventas, mejorando la eficiencia operativa y facilitando la toma de decisiones.

Evolución de los ERPs

Años 60 y 70: Aparición de los sistemas MRP (Planificación de Requerimientos de Materiales) enfocados en el control de inventarios y materiales.

Años 80: Evolución hacia los MRP II, incorporando la planificación de recursos de fabricación.

Años 90: Nacimiento de los ERP modernos, que abarcan una gama más amplia de procesos empresariales y adoptan arquitecturas cliente-servidor.

Siglo XXI: Transición hacia soluciones basadas en la nube, más flexibles y adaptadas a las necesidades dinámicas de las empresas.

Principales ERP:

SAP: Reconocido por su robustez y amplio alcance, ideal para grandes empresas.

Oracle NetSuite: Diseñado para negocios en crecimiento, con un enfoque en la nube.

Microsoft Dynamics 365: Combina funcionalidades de ERP y CRM, ofreciendo flexibilidad.

Odoo: Una alternativa de código abierto, modular y personalizable, adecuada para pequeñas y medianas empresas.

ERP seleccionado (Odoo):

Odoo ERP es el software libre de gestión empresarial capaz de cubrir todas las necesidades de tu negocio, gracias a la integración de sus múltiples aplicaciones: Odoo CRM, contabilidad, inventario, marketing online, gestión de proyectos, recursos humanos, etc. Odoo cuenta con una aplicación para cada necesidad empresarial, reuniendo en una única plataforma, tu solución personalizada, rentable y modular, lo que permitirá a tu organización ahorrar tiempo y recursos, gestionando de forma unificada tu negocio.

Instalación y desarrollo:

Instalación manual: Descargando e instalando los paquetes directamente en el sistema operativo.

Uso de Docker: Implementando contenedores que permiten una configuración rápida.

Plataformas en la nube: Utilizando Odoo online

Para este proyecto, se utilizará Docker debido a las siguientes razones:

Portabilidad: Los contenedores pueden ejecutarse en cualquier entorno compatible.

Facilidad de configuración: Simplifica el despliegue y la gestión de servicios.

Aislamiento: Garantiza que cada instancia de Odoo funcione de manera independiente, evitando conflictos.

Especificaciones técnicas:

Arquitectura de Odoo:

Base de datos: Utiliza PostgreSQL para gestionar información estructurada.

Servidor de aplicación: Desarrollado en Python, maneja la lógica del negocio y la interacción entre el cliente y la base de datos.

Cliente: Proporciona una interfaz web intuitiva para los usuarios.

Composición de un módulo:

Archivos Python: Definen la lógica y los modelos de datos.

Archivos XML: Configuran la parte visual de los módulos.

Archivos de seguridad: Gestionan permisos y accesos.

Archivos de datos: Incluyen información inicial, como plantillas y valores predeterminados.

Archivos de descripción: Contienen metadatos sobre el módulo, como su nombre, versión y dependencias.

2. SCRUM

Definición de SCRUM:

SCRUM es un marco de trabajo ágil utilizado para la gestión y ejecución de proyectos complejos. Este enfoque se basa en la entrega incremental y continua de productos o servicios mediante iteraciones conocidas como sprints. Su objetivo principal es maximizar la productividad y la adaptabilidad ante los cambios.

Evolución:

SCRUM se originó en el ámbito del desarrollo de software durante los años 90. Fue introducido por Ken Schwaber y Jeff Sutherland como una alternativa a los enfoques tradicionales de gestión de proyectos. Desde entonces, su aplicación se ha extendido a diversos sectores gracias a su capacidad para fomentar la colaboración, la flexibilidad y la mejora continua.

Funcionamiento

El funcionamiento de SCRUM se basa en un ciclo iterativo de planificación, ejecución y revisión. Cada ciclo, conocido como sprint, tiene una duración fija (generalmente entre 2 y 4 semanas) y está diseñado para entregar un incremento funcional del producto. Las etapas principales son:

Planificación del sprint: Definir los objetivos y las tareas que se completan en el sprint.

Ejecución del sprint: Trabajar en las tareas establecidas, manteniendo reuniones diarias para sincronizar avances.

Revisión del sprint: Evaluar el resultado del trabajo realizado y recopilar comentarios de los interesados.

Retrospectiva: Reflexionar sobre el proceso y proponer mejoras para los siguientes sprints.

Principales conceptos

SCRUM introduce una serie de conceptos fundamentales que estructuran su funcionamiento:

Proyecto: Conjunto de actividades dirigidas a crear un producto o servicio con un valor claro para los interesados.

Historias de usuario: Descripciones breves y simples de funcionalidades o requisitos desde la perspectiva del usuario final.

Sprint: Periodo de tiempo fijo durante el cual se realiza un conjunto de tareas para entregar un incremento del producto.

Tarea: Actividad específica y concreta que forma parte de una historia de usuario y debe completarse durante el sprint.

Eventos clave: Reuniones que estructuran el flujo de trabajo, como las planificaciones, reuniones diarias, revisiones y retrospectivas.

4- CONTINUACION PROYECTO MANAGE

Objetivos:

Con este proyecto se ha intentado recrear un entorno de trabajo de una empresa real con sus tareas, sprints y todo lo relacionado con el trabajo en equipo de una empresa. Hemos creado una base de datos relacional que gestiona todas las tareas, sprints, historias de usuarios, proyectos...

El entorno de trabajo para desarrollar este proyecto incluye diversas herramientas que permiten una gestión eficiente y organizada de las tareas. A continuación, se describen las principales:

Docker:

Docker es una plataforma de contenedores que facilita la creación, despliegue y ejecución de aplicaciones. Para este proyecto, Docker se utiliza para:

Crear un entorno aislado y reproducible.

Simplificar la instalación y configuración de Odoo.

Garantizar la portabilidad del entorno de desarrollo.

Navegador web:

El navegador web es una herramienta esencial para interactuar con la interfaz de usuario de Odoo. Se utiliza para:

Acceder al cliente web de Odoo.

Probar y validar las funcionalidades desarrolladas.

PyCharm:

PyCharm es un entorno de desarrollo integrado (IDE) especializado en Python. Lo uso por las siguientes razones:

Proporciona herramientas avanzadas para la escritura y depuración de código.

Facilita la navegación por el código fuente de Odoo y sus módulos.

Integra sistemas de control de versiones como Git, mejorando la colaboración.

Control de versiones (Git):

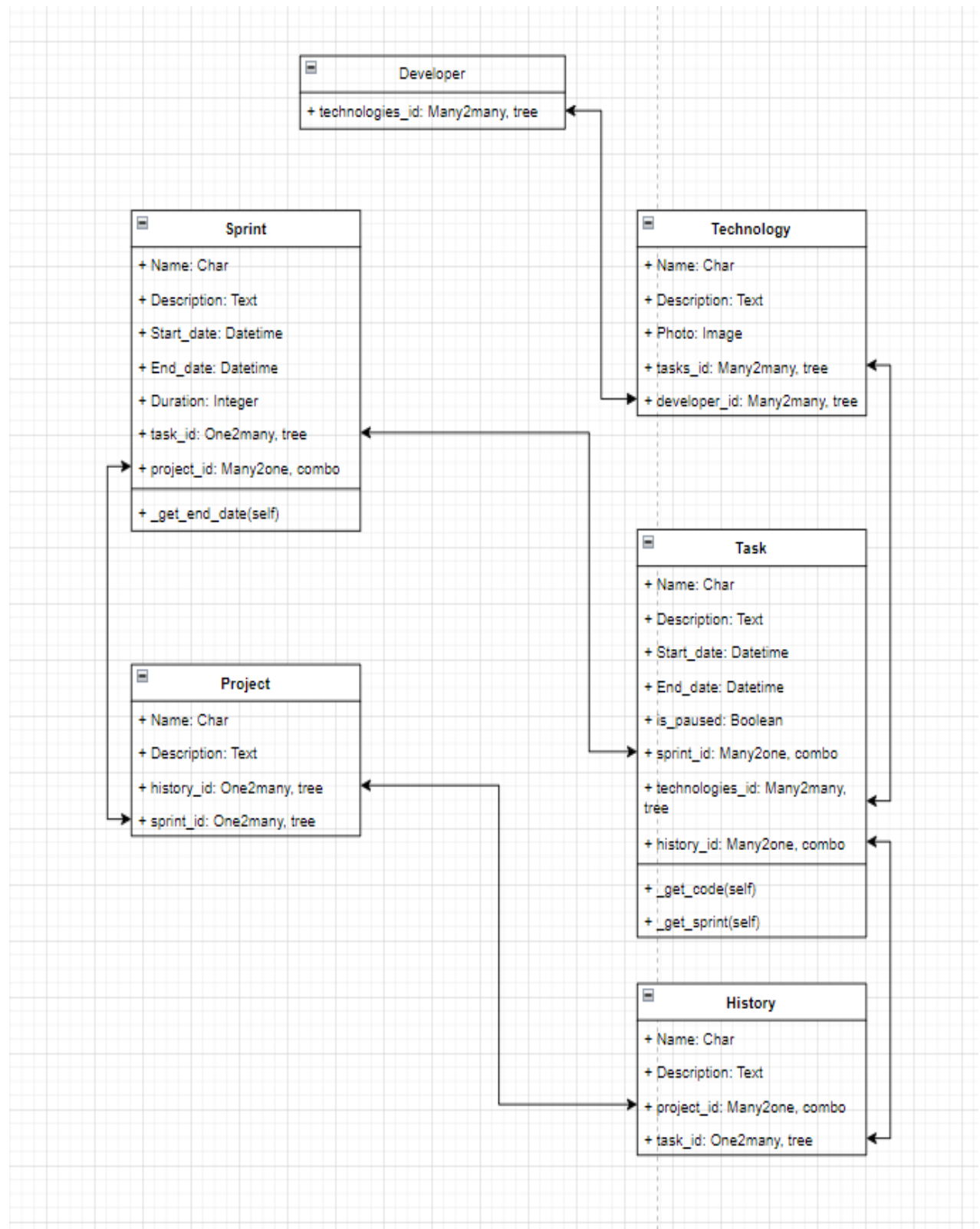
Se utiliza Git para gestionar los cambios en el código fuente del proyecto. Sus ventajas incluyen:

Rastrear el historial de modificaciones.

Facilitar la colaboración entre desarrolladores.

Permitir la recuperación de versiones anteriores en caso de errores.

5- Modelo relacional de la BBDD



6- Partes del proyecto

El proyecto está compuesto por varias partes fundamentales que permiten su correcto funcionamiento y organización. Estas son:

Models:

Los modelos son la base de los datos en Odoo. Definen las estructuras de las tablas en la base de datos y las relaciones entre ellas. Ejemplos de modelos en este proyecto incluyen:

Task: Gestiona las tareas con atributos como nombre, descripción, fechas de inicio y fin, entre otros.

```
class task(models.Model):
    _name = "manageisaac.task"
    _description = "manageisaac.task"

    code = fields.Char(string="Código", 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")
    is_paused = fields.Boolean(string="Pausada")
    definition_date = fields.Datetime(default=lambda p: datetime.datetime.now())

    sprint_id = fields.Many2one("manageisaac.sprint", string="Sprint", required=True, ondelete="cascade")
    sprint = fields.Many2one("manageisaac.sprint", compute="_get_sprint", store=True)
    technologies_id = fields.Many2many(comodel_name="manageisaac.technology", relation="technologies_areas", column1="technologies_ids", column2="tasks_ids")
    history_id = fields.Many2one("manageisaac.history", string="Historia", required=True, ondelete="cascade")
    project_id = fields.Many2one("manageisaac.project", string="Proyecto", related="history_id.project_id", ondelete="cascade", readonly=True)

    def _get_code(self):
        for task in self:
            if len(task.sprint_id) == 0:
                task.code = "TASK_"+str(task.id)
            else:
                task.code = str(task.sprint_id.name).upper()+"_"+str(task.id)

    @api.depends("code")
    def _get_sprint(self):
        for task in self:
            sprints = self.env["manageisaac.sprint"].search([("project_id.id", "=", task.history_id.project_id.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
```

Sprint: Representa los sprints y sus propiedades como nombre, descripción y duración.

```
class sprint(models.Model):
    _name = "manageisaac.sprint"
    _description = "manageisaac.sprint"

    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")
    duration = fields.Integer(string="Duración", default=15)

    task_id = fields.One2many(string="Tasks", comodel_name="manageisaac.task", inverse_name="sprint_id")
    project_id = fields.Many2one("manageisaac.project", string="Proyecto", required=True, ondelete="cascade")

    @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
```

Project: Contiene información sobre los proyectos y sus relaciones con sprints e historias.

```
class project(models.Model):
    _name = "manageisaac.project"
    _description = "manageisaac.project"

    name = fields.Char(string="Nombre", readonly=False, required=True, help="Introduzca el nombre")
    description = fields.Text(string="Descripción")

    history_id = fields.One2many(string="Historias", comodel_name="manageisaac.history", inverse_name="project_id")
    sprint_id = fields.One2many(string="Sprints", comodel_name="manageisaac.sprint", inverse_name="project_id")
    task_id = fields.One2many(string="Tareas", comodel_name="manageisaac.task", inverse_name="project_id")
```

Otras clases:

Controla todos los datos de las historias, está relacionada con las tareas, los proyectos y las tecnologías usadas.

```
class history(models.Model):
    _name = "manageisaac.history"
    _description = "manageisaac.history"

    name = fields.Char(string="Nombre", readonly=False, required=True, help="Introduzca el nombre")
    description = fields.Text(string="Descripción")

    project_id = fields.Many2one("manageisaac.project", string="Proyecto", required=True, ondelete="cascade")
    task_id = fields.One2many(string="Tareas", comodel_name="manageisaac.task", inverse_name="history_id")
    used_technologies = fields.Many2many("manageisaac.technology", compute="_get_used_technologies")

    def _get_used_technologies(self):
        for history in self:
            technologies = None #Array para concatenar todas las tecnologías
            for task in history.task_id: #para cada una de las tareas de la historia
                if not technologies:
                    technologies = task.technologies_id
                else:
                    technologies = technologies + task.technologies_id
            history.used_technologies = technologies #Asignar las tecnologías a la historia
```

Define los datos que tendrán las tecnologías usadas.

```
class technology(models.Model):
    _name = "manageisaac.technology"
    _description = "manageisaac.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="manageisaac.task", relation="technologies_tareas", column1="tasks_ids", column2="technologies_ids")
    #developer_id = fields.Many2many(comodel_name="manageisaac.developer", relation="developer_technologies", column1="technologies_id", column2="developer_id")
```

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

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

Views

Las vistas definen la interfaz de usuario en Odoo. Incluyen:

Form views: Permiten a los usuarios crear y editar los datos de los módulos.

```
<record model="ir.ui.view" id="vista_manageisaac_task_form">
  <field name="name">vista_manageisaac_task_form</field>
  <field name="model">manageisaac.task</field>
  <field name="arch" type="xml">
    <form string="formulario_task">
      <sheet>
        <group name="group_top">
          <field name="code"/>
          <field name="name"/>
          <field name="description"/>
          <field name="start_date"/>
          <field name="end_date"/>
          <field name="is_paused"/>
          <field name="definition_date"/>
          <field name="sprint_id"/>
          <field name="technologies_id"/>
          <field name="history_id"/>
          <field name="project_id"/>
        </group>
      </sheet>
    </form>
  </field>
</record>
```

Código									
Nombre ?									
Descripción									
Fecha de inicio									
Fecha de fin									
Pausada	<input type="checkbox"/>								
Definition Date	14/01/2025 08:35:57								
Sprint									
Technologies	<table><thead><tr><th>Nombre</th><th>Descripción</th></tr></thead><tbody><tr><td colspan="2">Agregar una línea</td></tr><tr><td colspan="2"></td></tr><tr><td colspan="2"></td></tr></tbody></table>	Nombre	Descripción	Agregar una línea					
Nombre	Descripción								
Agregar una línea									
Historia									
Proyecto									

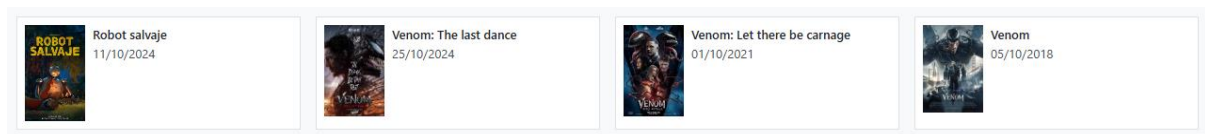
Tree views: Muestran listas de todos los objetos que hay de un módulo.

```
<record model="ir.ui.view" id="vista_manageisaac_task_tree">
  <field name="name">vista_manageisaac_task_tree</field>
  <field name="model">manageisaac.task</field>
  <field name="arch" type="xml">
    <tree>
      <field name="name"/>
      <field name="start_date"/>
      <field name="end_date"/>
      <field name="description"/>
    </tree>
  </field>
</record>
```

<input type="checkbox"/>	Nombre	Fecha de inicio
<input type="checkbox"/>	TAREA 1	30/12/2024 13:31:53
<input type="checkbox"/>	TAREA 2	14/01/2025 08:37:13
<input type="checkbox"/>	TAREA 3	09/01/2025 08:37:30

Kanban views: Representan los datos de manera visual y organizada.

```
<record model="ir.ui.view" id="vista_filmotecaisaac_pelicula_kanban">
  <field name="name">vista_filmotecaisaac_pelicula_kanban</field>
  <field name="model">filmotecaisaac.pelicula</field>
  <field name="arch" type="xml">
    <kanban>
      <field name="id"/>
      <field name="name"/>
      <field name="film_date"/>
      <templates>
        <t t-name="kanban-box">
          <div t-attf-class="oe_kanban_global_click">
            <div class="o_kanban_image">
              
            </div>
            <div class="o_kanban_details">
              <strong class="o_kanban_record_title">
                <field name="name"/>
              </strong>
              <div t-if="record.film_date.value">
                <t t-esc="record.film_date.value"/>
              </div>
            </div>
          </div>
        </t>
      </templates>
    </kanban>
  </field>
</record>
```



El archivo views.xml contiene ejemplos por defecto de alguna de estas vistas.

Security

La seguridad en Odoo se implementa mediante reglas de acceso y controles que aseguran que los usuarios solo puedan realizar acciones autorizadas. El archivo ir.model.access.csv define:

```
1 |id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 |access_manageisaac_task,manageisaac.task,model_manageisaac_task,base.group_user,1,1,1,1
3 |access_manageisaac_sprint,manageisaac.sprint,model_manageisaac_sprint,base.group_user,1,1,1,1
4 |access_manageisaac_project,manageisaac.project,model_manageisaac_project,base.group_user,1,1,1,1
5 |access_manageisaac_history,manageisaac.history,model_manageisaac_history,base.group_user,1,1,1,1
6 |access_manageisaac_technology,manageisaac.technology,model_manageisaac_technology,base.group_user,1,1,1,1
```

Grupos de usuarios: Quién tiene acceso a qué modelos.

Permisos: Acciones permitidas como leer, escribir, crear o eliminar registros.

Otros componentes

Archivos de datos: Configuraciones iniciales o valores predeterminados.

Controladores: Gestionan las interacciones personalizadas entre el cliente y el servidor.

El archivo manifest: tiene todos los datos del proyecto y hace que se vean las vistas en un orden específico, que el proyecto tenga seguridad...

```
No Python interpreter configured for the project Use Python 3.13 (Practica12_IsaacGo)
1  # -*- coding: utf-8 -*-
2  {
3      'name': "manageisaac",
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/developer.xml',
29         'views/technology.xml',
30         'views/project.xml',
31         'views/history.xml',
32         'views/sprint.xml',
33         'views/task.xml',
34         'views/views.xml',
35         'views/templates.xml',
36     ],
37     # only loaded in demonstration mode
38     'demo': [
39         'demo/demo.xml',
40     ],
41 }
```


7- POSIBLE AMPLIACIÓN

Una ampliación podría ser la añadir un nuevo campo a las tareas que sea de prioridad para que el trabajador que use la aplicación pueda priorizar unas tareas antes que otras e identificarlas de manera más fácil ya que en la vista tree se verían ordenadas por prioridad de cada una.

```
class task(models.Model):
    _name = "manageisaac.task"
    _description = "manageisaac.task"
    _order = "priority_order desc, end_date asc" # Ordena por prioridad y luego por fecha de fin solo si tienen la misma prioridad
```

```
priority = fields.Selection([('low', 'Baja'),('medium', 'Media'),('high', 'Alta')], string="Prioridad", required=True)
priority_order = fields.Integer(string="Orden de prioridad", compute="_compute_priority_order", store=True)
```

```
@api.depends("priority")
def _compute_priority_order(self):
    for task in self:
        if task.priority == 'low':
            task.priority_order = 1
        elif task.priority == 'medium':
            task.priority_order = 2
        elif task.priority == 'high':
            task.priority_order = 3
```

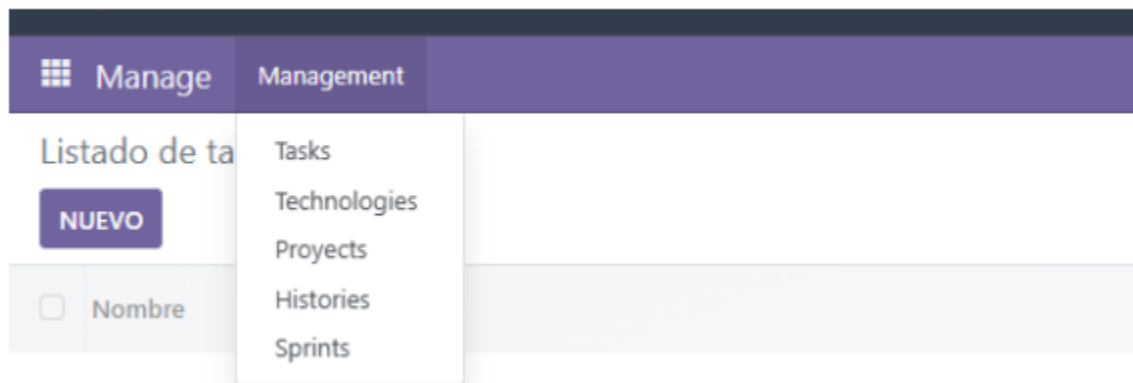
Con este código he conseguido que según creas una tarea y le pones una prioridad (alta, media o baja) se ordenan dependiendo de eso en la vista tree. Además, si varias tareas tienen la misma prioridad se ordenan por la fecha de fin. Esto puede ser muy útil en proyectos que tengan muchas tareas y se necesiten hacer unas antes que otras.

<input type="checkbox"/>	Nombre	Prioridad	Fecha de inicio	Fecha de fin
<input type="checkbox"/>	TAREA 5	Alta	14/01/2025 09:22:04	15/01/2025 09:22:04
<input type="checkbox"/>	TAREA 6	Alta	14/01/2025 09:32:01	15/01/2025 09:32:01
<input type="checkbox"/>	TAREA 2	Alta	29/01/2025 08:58:27	30/01/2025 08:58:27
<input type="checkbox"/>	TAREA 4	Media	14/01/2025 09:05:31	15/01/2025 09:05:31
<input type="checkbox"/>	TAREA 1	Media	14/01/2025 13:31:53	15/01/2025 13:31:53
<input type="checkbox"/>	TAREA 3	Baja	06/01/2025 09:05:40	07/01/2025 09:05:40

Código	SPRINT 1_11								
Nombre ?	TAREA 5								
Descripción									
Prioridad	Alta								
Fecha de inicio	Baja								
Fecha de fin	Media								
Pausada	<input type="checkbox"/>								
Fecha de definición	14/01/2025 09:22:03								
Sprint	SPRINT 1								
Technologies	<table><thead><tr><th>Nombre</th><th>Descripción</th></tr></thead><tbody><tr><td colspan="2">Agregar una línea</td></tr><tr><td colspan="2"></td></tr><tr><td colspan="2"></td></tr></tbody></table>	Nombre	Descripción	Agregar una línea					
Nombre	Descripción								
Agregar una línea									
Historia	HISTORIA 1								
Proyecto	PROYECTO 1								

8- PRUEBAS DE FUNCIONAMIENTO

Menú de la aplicación.



Para crear una tarea te pide el nombre, una descripción, las fechas de inicio y final, varias tecnologías aplicadas y requiere obligatoriamente un Sprint y una Historia.

Código ?									
Nombre ?	TAREA 1								
Descripción ?	TAREA 1								
Fecha de inicio ?	30/12/2024 13:31:53								
Fecha de fin ?	22/01/2025 13:31:53								
Pausada ?	<input type="checkbox"/>								
Definition Date ?	13/01/2025 13:31:53								
Sprint ?	SPRINT 1								
Technologies ?	<table><thead><tr><th>Crear "SPRINT 1"</th><th>Descripción</th></tr></thead><tbody><tr><td colspan="2">Crear y editar...</td></tr><tr><td colspan="2"></td></tr><tr><td colspan="2"></td></tr></tbody></table>	Crear "SPRINT 1"	Descripción	Crear y editar...					
Crear "SPRINT 1"	Descripción								
Crear y editar...									
Historia ?									
Proyecto ?									

Para crear un Sprint, nombre, descripción, duración, fechas se le puede agregar varias tareas y un proyecto obligatorio.

Crear Sprint

Nombre ?

SPRINT 1

Descripción ?

SPRINT 1

Duración ?

15

Fecha de inicio ?

30/12/2024 13:33:20

Fecha de fin ?

30/01/2025 13:33:20

Tasks ?

Nombre	Fecha de inicio	Fecha de fin	Descripción	
Agregar una línea				

Proyecto ?

PROYECTO 1

Crear "PROYECTO 1"

Crear y editar...

GUARDAR Y CERRAR

DESCARTAR

Para crear una tecnología se introduce el nombre, la descripción, se le añade una o varias tareas y una foto.

Nombre ?

TECNOLOGIA 1

Descripción ?

TECNOLOGIA 1

Tasks ?

Nombre	Fecha de inicio	Fecha de fin	Descripción	
TAREA 1	30/12/2024 13:31:53	22/01/2025 13:31:53	TAREA 1	✕
Agregar una línea				

Imagen ?

Para crear un proyecto se le pone nombre, descripción, una o varias historias y uno o varios sprints.

Nombre [?]

PROYECTO 1

Descripción [?]

PROYECTO 1

Historias [?]

Nombre	Descripción
HISTORIA 1	HISTORIA 1
Agregar una línea	

Sprints [?]

Nombre	Descripción	Fecha de inicio	Fecha de fin
SPRINT 1	SPRINT 1	30/12/2024 13:33:20	30/01/2025 13:33:20
Agregar una línea			

Y si vas a la pestaña de developers te saldrá una lista con todos los que hay. Si entras en uno te sale toda la información suya.

Nombre	Teléfono	Correo electrónico	Comercial	Actividades	Ciudad	País	Compañía
<input type="checkbox"/> Azure Interior	+58 212 681 0538	vauxoo@yourcompany.example.com	Comercial	<input type="radio"/>	Fremont	Estados Unidos	
<input type="checkbox"/> Azure Interior, Brandon Freeman	(355) 687-3262	brandon.freeman55@example.com		<input type="radio"/>	Fremont	Estados Unidos	
<input type="checkbox"/> Azure Interior, Colleen Diaz	(255) 595-8393	colleen.diaz83@example.com		<input type="radio"/>	Fremont	Estados Unidos	
<input type="checkbox"/> Azure Interior, Nicole Ford	(946) 638-6034	nicole.ford75@example.com		<input type="radio"/>	Fremont	Estados Unidos	
<input type="checkbox"/> Devo Adick	+1 312 349 2324	john@thetechinfo.com		<input type="radio"/>	Peasant Hill	Estados Unidos	
<input type="checkbox"/> Devo Adick, Addison Olson	(225) 555-7037	addison.olson23@example.com		<input type="radio"/>	Peasant Hill	Estados Unidos	
<input type="checkbox"/> Devo Adick, Douglas Fletcher	(133) 553-7142	douglas.fletcher1@example.com		<input type="radio"/>	Peasant Hill	Estados Unidos	
<input type="checkbox"/> Devo Adick, Floyd Steward	(140) 555-9401	floyd.steward24@example.com		<input type="radio"/>	Peasant Hill	Estados Unidos	
<input type="checkbox"/> FFI Company	+1 415 514 56 76	info@regency-spa.example.com		<input type="radio"/>	Concordia	Francia	
<input type="checkbox"/> Gemini Furniture	+1 312 349 2324	john@thetechinfo.com		<input type="radio"/>	Fairfield	Estados Unidos	
<input type="checkbox"/> Gemini Furniture, Eileen Hansen	(840) 555-5255	eileen.hansen75@example.com		<input type="radio"/>	Fairfield	Estados Unidos	
<input type="checkbox"/> Gemini Furniture, Jesse Brown	(826) 309-3277	jesse.brown1@example.com		<input type="radio"/>	Fairfield	Estados Unidos	My Company (San Francisco)
<input type="checkbox"/> Gemini Furniture, Oscar Morgan	(261) 535-1744	oscar.morgan1@example.com		<input type="radio"/>	Fairfield	Estados Unidos	
<input type="checkbox"/> Gemini Furniture, Sabine Palmer	(376) 367-5940	sabine.palmer1@example.com		<input type="radio"/>	Fairfield	Estados Unidos	
<input type="checkbox"/> Lumber Inc.	(826) 210-0592	lumber-info@example.com		<input type="radio"/>	Bedford	Estados Unidos	
<input type="checkbox"/> Lumber Inc, Lorraine Douglas	(440) 646-9155	lorraine.douglas23@example.com		<input type="radio"/>	Bedford	Estados Unidos	
<input type="checkbox"/> My Company Chicago	+1 312 349 2324	chicago@yourcompany.com		<input type="radio"/>	Chicago	Estados Unidos	
<input type="checkbox"/> My Company Chicago, Jeff Lawson	(401) 417-6287	jeff.lawson52@example.com		<input type="radio"/>	Chicago	Estados Unidos	
<input type="checkbox"/> My Company San Francisco	+1 555 555 5556	info@yourcompany.com		<input type="radio"/>	San Francisco	EEUU	
<input type="checkbox"/> My Company San Francisco, Chester Reed	(870) 904-6932	chester.reed79@example.com		<input type="radio"/>	San Francisco	EEUU	
<input type="checkbox"/> My Company San Francisco, Douglas Neumann	(816) 577-4837	douglas.neumann79@example.com		<input type="radio"/>	San Francisco	Francia	

2 Reuniones

2 Oportunida...


0 Ventas

31.750,00 € Facturado

Individuo

Compañía

Azure Interior



Dirección

4557 De Silva St

Calle 2...

California (US)

Estados Unidos

NIF [?]

p. ej., ESA00000000

Teléfono [?]

+58 212 681 0538

Móvil [?]

Correo electrónico [?]

vauxoo@yourcompany.example.com

Sitio web [?]

http://www.azure-interior.com

Etiquetas [?]

Servicios ✖


Contactos y direcciones

Ventas y compras


Facturación / Contabilidad

Notas internas


AGREGAR



Brandon Freeman
Creative Director
brandon.freeman55@example.com
Teléfono: (355)-687-3262



Colleen Diaz
Business Executive
colleen.diaz83@example.com
Teléfono: (255)-595-8393



Nicole Ford
Director
nicole.ford75@example.com
Teléfono: (946)-638-6034


Enviar mensaje

Registrar una nota

Actividades

Seguir

21 de octubre de 2024

 OdooBot · hace 3 meses

- azure.interior24@example.com → vauxoo@yourcompany.example.com (Email)
- (870)-931-0505 → +58 212 681 0538 (Phone)

{ 20 }

9- CONCLUSIONES

El proyecto ha dejado claro cómo usar Odoo y SCRUM puede mejorar los procesos y a gestionar proyectos de mejor forma. Al final, he conseguido crear un sistema funcional que cumple con los objetivos y que se puede mejorar o ampliar. También usé PyCharm, lo que me ayudó a tener un entorno de desarrollo más ordenado y eficiente ya que su interfaz es de inteliJ, que es más fácil entender cómo hacer las cosas que en VisualStudio.

10- BIBLIOGRAFÍA

- <https://www.sap.com/spain/products/erp/what-is-erp.html>
- <https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/>
- <https://www.atlassian.com/es/agile/scrum>
- <https://www.indaws.es/que-es-odoo>
- <https://www.wolterskluwer.com/es-es/expert-insights/que-es-un-software-erp-tipos-y-ejemplos>
- <https://asana.com/es/resources/what-is-scrum>