



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: 22/12/2024

Nombre y Apellidos: Leire Yagüe Fernández
Email: leire.yagfer.1@educa.jcyl.es

ÍNDICE

1	INTRODUCCIÓN	3
2	ESTADO DEL ARTE.....	3
2.1	ERP	3
2.1.1	DEFINICIÓN DE LOS ERP	3
2.1.2	EVOLUCIÓN DE LOS ERPS	3
2.1.3	PRINCIPALES ERP	4
2.1.4	ERP SELECCIONADO (ODOO).....	4
2.1.5	INSTALACIÓN Y DESARROLLO (FORMAS DE INSTALACIÓN, EXPLICANDO LA QUE SE VA A USAR: DOCKER) 4	
2.1.6	ESPECIFICACIONES TÉCNICAS.....	5
2.2	SCRUM.....	6
2.2.1	DEFINICIÓN DE SCRUM	6
2.2.2	EVOLUCIÓN	6
2.2.3	FUNCIONAMIENTO.....	6
2.2.4	PRINCIPALES CONCEPTOS (explicar los principales conceptos: proyecto, historias de usuario, sprint, tarea...)	7
3	DESCRIPCIÓN GENERAL DEL PROYECTO.....	8
3.1	OBJETIVOS	8
3.2	ENTORNO DE TRABAJO (explicar las herramientas utilizadas para desarrollar el proyecto: Docker, navegador, visual studio code...).....	8
3.3	DISEÑO DE LA APLICACIÓN.....	9
3.3.1	MODELO RELACIONAL DE LA BBDD.....	9
3.3.2	PARTES DEL PROYECTO	10
3.3.3	ORGANIZACIÓN DEL PROYECTO	10
4	PRUEBAS DE FUNCIONAMIENTO	23
5	AMPLIACIÓN.....	29
5.1	BOTÓN DE ELIMINAR TAREAS	29
5.2	VISTA KANBAN.....	29
6	CONCLUSIONES Y POSIBLES AMPLIACIONES	30
7	ENLACE AL PROYECTO MANAGELEIRE	31
8	BIBLIOGRAFÍA	32

1 INTRODUCCIÓN

Un **sistema ERP** (Enterprise Resource Planning) es una solución de software que permite integrar y gestionar las operaciones clave de una empresa, como finanzas, recursos humanos, ventas y producción, entre otras, en una única plataforma. Su principal ventaja es la centralización de los datos, lo que facilita la toma de decisiones, mejora la eficiencia y reduce los errores. Ejemplos de ERP populares son SAP, Oracle y Microsoft Dynamics 365.

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto. Se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

2 ESTADO DEL ARTE

2.1 ERP

2.1.1 DEFINICIÓN DE LOS ERP

Un ERP (Enterprise Resource Planning) es un software que integra y gestiona las operaciones clave de una empresa en una sola plataforma, centralizando datos para mejorar la eficiencia, reducir errores y facilitar la toma de decisiones. Ejemplos: SAP, Oracle y Microsoft Dynamics 365.

2.1.2 EVOLUCIÓN DE LOS ERPS

- **1960:** nacen los MRP en la industria manufacturera para gestionar inventarios y niveles de stock.
- **1970:** IBM difunde los MRP, planificando materias primas con grandes y costosos ordenadores centrales.
- **1980:** evolución al MRP-II, integrando inventarios y aspectos financieros como costos de materia prima y mano de obra.
- **1990:** surge el ERP moderno, con sistemas modulares para todas las áreas empresariales.
- **2000:** con internet, aparece el ERP de segunda generación, integrando CRM y otras herramientas.
- **2010:** ERP en la nube (SaaS) reduce costos y se populariza entre PyMEs.
- **Actualidad:** sistemas en la nube como Calipso permiten modelización y soluciones personalizadas de extremo a extremo.

2.1.3 PRINCIPALES ERP

Existen diversos sistemas ERP adaptados a distintos tamaños y sectores empresariales. Algunos de los más destacados:

- SAP ERP: para grandes corporaciones, con amplia personalización.
- Oracle ERP Cloud: solución en la nube, centrada en escalabilidad e integración tecnológica.
- Microsoft Dynamics 365: integrado con Office, ideal para medianas y grandes empresas.
- Odoo: ERP de código abierto, modular y accesible, ideal para pymes.
- Infor ERP: enfocado en industrias específicas, con énfasis en la experiencia del usuario y movilidad.

2.1.4 ERP SELECCIONADO (ODOO)

Odoo es un ERP de software integrado completamente abierto basado en lenguaje Python, que integra un sinfín de aplicaciones y módulos de gestión empresarial, además de integrar un sitio web y tienda online que puede ser modificado y adaptado según las necesidades y exigencias del cliente.

Características principales de Odoo:

- Modularidad: más de 30 aplicaciones para áreas clave, con módulos adicionales de la comunidad.
- Accesibilidad: disponible en la nube o en servidores locales.
- Interfaz amigable: fácil de usar, reduce el tiempo de adopción.
- Costo-efectividad: modelo de código abierto, pagando solo por los módulos necesarios.
- Actualizaciones constantes: integración de tecnologías modernas para mantener el sistema actualizado.

Ventajas de Odoo:

- Flexibilidad: adaptable y personalizable según las necesidades de la empresa.
- Comunidad activa: soporte continuo y nuevos desarrollos.
- Escalabilidad: posibilidad de empezar con pocos módulos y ampliar según lo necesario.

2.1.5 INSTALACIÓN Y DESARROLLO (FORMAS DE INSTALACIÓN, EXPLICANDO LA QUE SE VA A USAR: DOCKER)

Formas de Instalación de Odoo:

1. **Instalación en Servidores Locales (Bare Metal):** Odoo se instala en un servidor físico, requiriendo configuración manual de servicios como bases de datos y servidor web. Es adecuado para usuarios avanzados y empresas con necesidades específicas.
2. **Instalación en la Nube:** Odoo se implementa en servidores en la nube (AWS, Google Cloud, Azure), facilitando el acceso remoto y la gestión centralizada, pero con costos adicionales de infraestructura.
3. **Instalación a través de Docker (Método Seleccionado para el Proyecto):** Odoo se ejecuta en contenedores virtuales, garantizando un entorno consistente y controlado. Este método facilita la portabilidad, escalabilidad y la creación de entornos de desarrollo y producción sin problemas de configuración.

Instalación de Docker:

Para desarrollar el proyecto utilizando Docker, se realiza la instalación de Odoo en un contenedor que contiene todas las dependencias necesarias, como la base de datos PostgreSQL y los módulos requeridos por la aplicación. A continuación, se describe el proceso básico de instalación y desarrollo con Docker:

- Requisitos Previos:
 - Tener instalado Docker en la máquina. Docker se puede descargar desde docker.com.
 - Tener Docker Compose instalado, que es una herramienta que permite definir y ejecutar aplicaciones con múltiples contenedores de Docker.
- Pasos de Instalación:
 1. Clonar el repositorio oficial de Odoo o crear un archivo docker-compose.yml personalizado. El archivo docker-compose.yml define los servicios que Docker debe ejecutar, como Odoo y PostgreSQL.
 2. Ejecutar el comando para iniciar los contenedores con Docker Compose: “docker-compose up -d”.
 3. Una vez que los contenedores están en ejecución, se puede acceder a Odoo a través de un navegador web, accediendo a la URL <http://localhost:8069>, el puerto depende del que se haya puesto en Docker-compose.yml.

2.1.6 ESPECIFICACIONES TÉCNICAS

2.1.6.1 Arquitectura de Odoo

La **arquitectura de Odoo** se basa en una estructura de cliente-servidor, donde el cliente web interactúa con el servidor de aplicaciones, que se comunica con la base de datos PostgreSQL. La modularidad es uno de los puntos clave de Odoo, permitiendo personalizaciones y ampliaciones fáciles mediante módulos. El uso de Python como lenguaje de programación y PostgreSQL como base de datos garantiza una arquitectura sólida, escalable y eficiente.

2.1.6.2 Composición de un módulo

Un **módulo de Odoo** puede incluir los siguientes elementos, pero no todos los módulos contienen todos estos elementos. Algunos pueden incluir solo datos o solo objetos de negocio.

- **Objetos de negocio:** clases de Python que representan entidades, como una factura. Los campos de estas clases se vinculan automáticamente con las columnas de la base de datos usando el ORM.
- **Vistas de objetos:** definen cómo se presenta la información en la interfaz de usuario.
- **Archivos de datos:** archivos XML o CSV que configuran datos para el modelo, como configuraciones, reglas de seguridad o datos de demostración.
- **Controladores Web:** manejan las solicitudes del navegador web.
- **Datos Web estáticos:** imágenes, archivos CSS o JavaScript utilizados en la interfaz web.

2.2 SCRUM

2.2.1 DEFINICIÓN DE SCRUM

Scrum es un proceso que aplica buenas prácticas para trabajar en equipo y obtener el mejor resultado posible, realizando entregas parciales y regulares priorizadas según el beneficio para el receptor. Es ideal para proyectos en entornos complejos, con requisitos cambiantes, donde la innovación, flexibilidad y productividad son clave.

2.2.2 EVOLUCIÓN

- **1980:** origen de Scrum, adoptando su forma actual en 1995 cuando Ken Schwaber y Jeff Sutherland publicaron un artículo conjunto para hacer el trabajo más productivo y con el menor número de reglas posibles.
- **2010 – primera guía oficial de Scrum:** se reduce la literatura y se presenta Scrum como un "marco de trabajo" en lugar de una metodología.
- **2011 – definición de Scrum, grooming y descripción de roles:** Scrum se define como un framework para resolver problemas complejos y adaptativos. El grooming, proceso de revisión y actualización periódica del proyecto que se está realizando, se convierte en una práctica para preparar historias de usuario.
- **2013 – las tres preguntas en la Daily Scrum y el foco en el Sprint Goal:** se hace hincapié en la relevancia del valor en Scrum y se modifica la pregunta del Daily Scrum a “¿Qué hiciste ayer para ayudar a conseguir el Sprint Goal?”.
- **2016 – los valores de Scrum:** se introducen valores como compromiso, apertura, foco, respeto y coraje, necesarios para el éxito del equipo Scrum. La transparencia se convierte en una condición esencial.
- **2017 – las preguntas en el Daily Scrum son optativas y los usos de Scrum:** se reconocen los diferentes usos de Scrum más allá de la tecnología, y se eliminan prácticas innecesarias para centrarse en los conceptos clave.
- **2020 – autogestión, Product Goal y Scrum más allá del software:** se introducen nuevos cambios, incluyendo el enfoque en la autogestión y el Product Goal, reflejando la evolución de Scrum.

2.2.3 FUNCIONAMIENTO

Desarrollo del proceso Scrum:

1. **Organización del trabajo pendiente:** el Scrum Master identifica el trabajo que debe realizarse a partir de la lista de tareas pendientes, asegurándose de que esté claramente documentado en un solo lugar.
2. **Sprint Planning:** se realiza una sesión de planificación del sprint para evaluar qué parte del trabajo pendiente se abordará durante el sprint.
3. **Comienzo del sprint:** el sprint dura típicamente dos semanas, durante las cuales el equipo trabaja en las tareas seleccionadas en la planificación.
4. **Daily Stand Up:** reuniones diarias de 15 minutos para que el equipo informe sobre el progreso y posibles obstáculos, enfocándose en el trabajo del día siguiente.
5. **Sprint Review:** al final del sprint, se presenta el trabajo terminado para su revisión y aprobación.
6. **Sprint Retrospective:** análisis de cómo se desarrolló el sprint y qué se puede mejorar en el próximo.

Artefactos Scrum:

- **Product Backlog:** lista de trabajo pendiente del producto, organizada y gestionada por el Product Owner, que incluye tareas que pueden ser abordadas en próximos sprints.
- **Sprint Backlog:** lista de tareas seleccionadas para el sprint actual, extraídas del Product Backlog durante la planificación del sprint.
- **Incremento del Producto:** trabajo entregado al final de cada sprint, que puede incluir nuevas funciones, mejoras o correcciones.

Roles de Scrum:

- **Product Owner:** responsable de la lista de trabajo pendiente del producto, que transmite las necesidades del usuario y decide cuándo algo está listo para ser entregado.
- **Scrum Master:** facilita los eventos de Scrum, promoviendo reuniones diarias y organizando las reuniones de planificación, revisión y retrospectiva.
- **Equipo Scrum:** miembros autoorganizados que colaboran para lograr los objetivos del sprint.

Definición de "**Terminado**": cada equipo debe tener claro lo que significa "Terminado" (por ejemplo, listo para su lanzamiento, probado, aceptado), y esta definición debe mantenerse en referencia durante todo el proceso Scrum.

Burn Down y Burn Up:

- **Burn Down:** gráfico que muestra el trabajo pendiente durante el sprint, disminuyendo con el tiempo.
- **Burn Up:** gráfico que muestra el trabajo completado en el sprint.

Principios de Scrum:

1. **Control sobre el proceso empírico:** transparencia, inspección y adaptación.
2. **Autoorganización:** cada miembro del equipo es responsable de sus tareas, fomentando la creatividad y dinamismo.
3. **Colaboración: trabajar en conjunto para obtener los mejores resultados.**
4. **Priorización basada en valores:** entregar el mayor valor comercial priorizando tareas desde el inicio.
5. **Duración limitada (timeboxing):** actividades con tiempo limitado para asegurar la mejora continua.
6. **Desarrollo iterativo:** construir de manera iterativa para adaptarse a las necesidades del cliente y mejorar el producto según la priorización del valor.

2.2.4 PRINCIPALES CONCEPTOS (explicar los principales conceptos: proyecto, historias de usuario, sprint, tarea...)

- **Proyecto:** trabajo general que el equipo va a desarrollar, ya sea un producto o servicio. A diferencia de los proyectos tradicionales, que suelen planificarse completamente desde el principio, en SCRUM el proyecto es flexible y evolutivo. El proyecto se define a través de las funcionalidades y objetivos que se quieren lograr. El desarrollo se organiza en sprints, lo que permite entregar valor de manera continua y ajustarse a cambios en las necesidades del cliente.

- **Sprints:** SCRUM divide el trabajo en ciclos cortos y repetitivos llamados sprints, que suelen durar de 1 a 4 semanas. En cada sprint, el equipo se enfoca en completar un conjunto de tareas definidas para entregar una versión funcional del producto. El objetivo es entregar un producto funcional al final de cada sprint, que puede ser incrementado o ajustado en futuros sprints.
- **Historia de usuario:** manera en que SCRUM representa los requisitos o funcionalidades del proyecto desde la perspectiva del usuario final. Normalmente se escriben en el siguiente formato: "Como [tipo de usuario], quiero [funcionalidad], para [lograr un objetivo]". Ejemplo: "Como usuario registrado, quiero poder cambiar mi contraseña, para mejorar la seguridad de mi cuenta". El propósito es ayudar a enfocarse en las necesidades y beneficios reales para el usuario, destacando lo que realmente importa. Cada historia de usuario debe ser clara, pero no extremadamente detallada, lo que deja espacio para el diálogo entre los miembros del equipo y permite generar soluciones de manera colaborativa.
- **Tarea:** división más pequeña de trabajo que deriva de una historia de usuario. El equipo descompone las historias de usuario elegidas para el sprint en tareas específicas. Cada tarea representa un paso concreto necesario para completar la historia de usuario. Las tareas suelen ser asignadas y gestionadas por los miembros del equipo durante el sprint. Al ser de menor tamaño, las tareas permiten que el equipo gestione mejor su avance, detecte problemas rápidamente y mantenga un control eficiente del progreso. Cada tarea debe ser específica y realizable en unas pocas horas o como máximo en un día.

Un proyecto es la suma de las historias de usuario, que se completan a través de sprints.

Una historia tiene varias tareas asociadas.

Un sprint tiene varias historias; por lo cual, un sprint va a tener varias tareas asociadas.

En las tareas se van a usar diferentes tecnologías.

3 DESCRIPCIÓN GENERAL DEL PROYECTO

3.1 OBJETIVOS

El objetivo de la práctica es desarrollar el módulo "manageleire" desde cero con Odoo ERP para gestionar proyectos utilizando metodologías ágiles, específicamente SCRUM, con Python. Se implementaron metodologías como campos relacionales computados, campos con valores por defecto y herencia, entre otros.

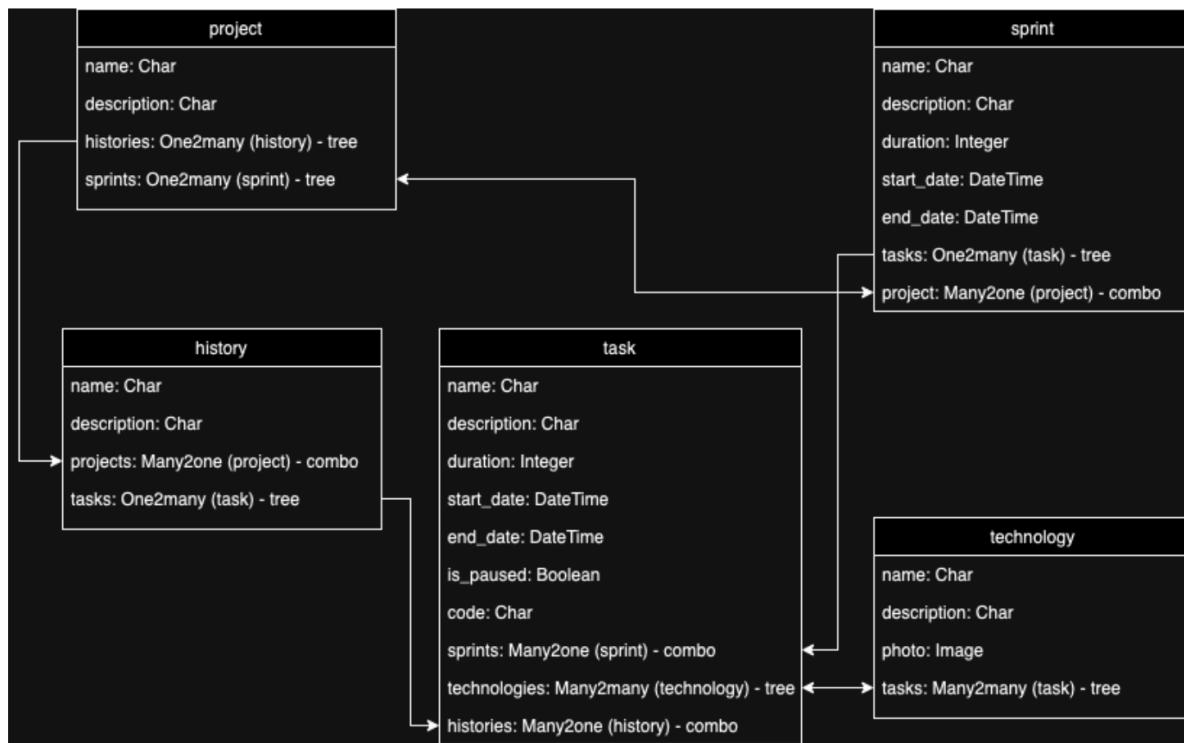
3.2 ENTORNO DE TRABAJO (explicar las herramientas utilizadas para desarrollar el proyecto: Docker, navegador, visual studio code...)

- **Docker:** usado para crear contenedores que aislan el entorno de desarrollo y facilitan la instalación y configuración de Odoo ERP. Esto permitió ejecutar el sistema de manera controlada y sin interferencias con el sistema operativo subyacente. Además, los contenedores garantizaron consistencia entre los distintos entornos de desarrollo, pruebas y producción.

- **Navegador:** Google Chrome fue el navegador principal utilizado para interactuar con la interfaz web de Odoo ERP. Además de su fiabilidad, Chrome ofrece herramientas de desarrollo como el inspector de elementos, que facilitaron la depuración de la interfaz de usuario y mejoraron la experiencia del usuario durante las pruebas del módulo.
- **Visual Studio Code:** utilizado como editor de código para el desarrollo en Python. Su ligereza y extensibilidad permitieron escribir, depurar y ejecutar código de manera eficiente. Además, su integración con Git ayudó en el control de versiones, asegurando un manejo adecuado del código fuente durante todo el proceso de desarrollo del módulo.

3.3 DISEÑO DE LA APLICACIÓN

3.3.1 MODELO RELACIONAL DE LA BBDD



3.3.2 PARTES DEL PROYECTO

3.3.3 ORGANIZACIÓN DEL PROYECTO

```
✓ MANAGELEIRE      ⌂ ⌂ ⌂ ⌂
  > __pycache__
  > controllers
  > demo
  > models
  > security
  > views
  ✎ __init__.py
  ✎ __manifest__.py
```



```
✓ MANAGELEIRE
  > __pycache__
  > controllers
  > demo
  ✓ models
    > __pycache__
    ✎ __init__.py
    ✎ developer.py
    ✎ history.py
    ✎ models.py
    ✎ project.py
    ✎ sprint.py
    ✎ task.py
    ✎ technology.py
  ✓ security
    ✎ ir.model.access.csv
  ✓ views
    ✎ developer.xml
    ✎ history.xml
    ✎ project.xml
    ✎ sprint.xml
    ✎ task.xml
    ✎ technology.xml
    ✎ templates.xml
    ✎ views.xml
    ✎ __init__.py
    ✎ __manifest__.py
```

3.3.3.1 MODELS

- `__init__.py`

```
models > __init__.py
1  # -*- coding: utf-8 -*-
2
3  from . import models
4  from . import task
5  from . import sprint
6  from . import project
7  from . import history
8  from . import technology
9  from . import developer
```

- `Developer`

```
models > developer.py > ...
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api # type: ignore
4
5  #TEMA 9 --> HERENCIA EN ODOO
6  class developer(models.Model):
7
8      ...
9
10     Con esto, estamos modificando la tabla de res.partner y añadiendo los campos que indiquemos (si
11     añadimos un campo ya existente, se va a sobreescribir).
12     ...
13     _name = 'res.partner'
14     _inherit = 'res.partner' #herencia
15
16     is_dev = fields.Boolean()
17     ...
18
19     Vamos a añadir campos a la tabla res.partner: technologies (Many2many). Este campo se va a
20     añadir al modelo res.partner y no se va a crear un modelo nuevo developer:
21     ...
22
23     technologies = fields.Many2many('managelire.technology',
24                                     relation='developer_technologies',
25                                     column1='developer_id',
26                                     column2='technologies_id')
```

- History

```
models > ⚡ history.py > ...
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api # type: ignore
4
5  #se va a traducir en una tabla en la base de datos
6  class history(models.Model):
7
8      #ATRIBUTOS
9      _name = 'manageleire.history' #nombreModulo.nombreModelo --> así le llamo desde Odoo
10     _description = 'manageleire.history'
11
12
13     #CAMPOS
14     #entre paréntesis es el nombre de la tabla que se va a mostrar en la vista
15     name = fields.Char(string = "Nombre de la historia", readonly = False, required = True, help = "Introduzca el nombre de la historia")
16             #readonly (solo lectura): false --> se puede editar. Si fuese true no se podría editar
17             #required: True --> obligatorio
18             #cuando estoy sobre Nombre en la vista, se ve el mensaje de help
19     description = fields.Char(string = "Descripción")
20
21
22     #RELACIONES
23     #Muchas historias pertenece a un proyecto
24     project_id = fields.Many2one(comodel_name="manageleire.project",
25                                     string="Proyecto",
26                                     required=True,
27                                     ondelete="cascade")
28
29     #Cada historia tiene muchas tareas
30     task_id = fields.One2many(string="Tareas",
31                               comodel_name="manageleire.task",
32                               inverse_name="history_id")
33
34     #TEMA 9
35     #Conjunto de tecnologías usadas en una historia
36     used_technologies = fields.Many2many(string="Tecnologías usadas",
37                                         comodel_name="manageleire.technolog",
38                                         compute="_get_used_technologies")
39
40
41     #FUNCIONES
42     #Comprobar todas las tecnologías utilizadas en las tareas de una historia y asignarlas al campo used_technologies de esa historia
43     def _get_used_technologies(self):
44         for history in self:
45             technologies = None #array para almacenar todas las tecnologías
46             for task in history.task_id: #recorrer todas las tareas de una historia
47                 if not technologies:
48                     technologies = task.tecnologias_id
49                 else:
50                     technologies = technologies + task.tecnologias_id
51             history.used_technologies = technologies #asignar las tecnologías a la historia
```

- Project

```
models > ⚡ project.py > ...
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api # type: ignore
4
5  #se va a traducir en una tabla en la base de datos
6  class project(models.Model):
7
8      #ATRIBUTOS
9      _name = 'manageleire.project' #nombreModulo.nombreModelo --> así le llamo desde Odoo
10     _description = 'manageleire.project'
11
12
13     #CAMPOS
14     #entre paréntesis es el nombre de la tabla que se va a mostrar en la vista
15     name = fields.Char(string = "Nombre del proyecto", readonly = False, required = True, help = "Introduzca el nombre del proyecto")
16             #readonly (solo lectura): false --> se puede editar. Si fuese true no se podría editar
17             #required: True --> obligatorio
18             #cuando estoy sobre Nombre en la vista, se ve el mensaje de help
19     description = fields.Char(string = "Descripción")
20
21
22     #RELACIONES
23     #Un proyecto tiene varias historias
24     history_id = fields.One2many(string="Historias",
25                                   comodel_name="manageleire.history",
26                                   inverse_name="project_id")
27
28     #Un proyecto tiene muchos sprints
29     sprint_id = fields.One2many(string="Carreras",
30                                 comodel_name="manageleire.sprint",
31                                 inverse_name= "project_id")
```

- Sprint

```

models > sprint.py > sprint
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api # type: ignore
4  import datetime
5
6  #se va a traducir en una tabla en la base de datos
7  class Sprint(models.Model):
8
9      #ATRIBUTOS
10     _name = 'manageleire.sprint' #nombreModulo.nombreModelo --> así le llamo desde Odoo
11     _description = 'manageleire.sprint'
12
13
14      #CAMPOS
15      #entre paréntesis es el nombre de la tabla que se va a mostrar en la vista
16      name = fields.Char(string = "Nombre del sprint", readonly = False, required = True, help = "Introduzca el nombre del sprint")
17          #readonly (solo lectura): false --> se puede editar. Si fuese true no se podría editar
18          #required: True --> obligatorio
19          #cuando estoy sobre Nombre en la vista, se ve el mensaje de help
20      description = fields.Char(string = "Descripción")
21      start_date = fields.Datetime(string = "Fecha inicio")
22      #el compute y store es debido al uso de la función abajo definida (_get_end_date)
23      end_date = fields.Datetime(string = "Fecha fin", compute=_get_end_date, store=True)
24
25      #PRACTICA 17 - TEMA 8 --> con la fecha de inicio y la duración, calcularemos la fecha de fin. Además, la fecha de fin se almacena
26      duration = fields.Integer(string="Duración (días)", default=15)
27
28
29      #RELACIONES
30      #Cada sprint tiene multiples tareas asignadas; cada tarea se asigna a un sprint específico
31      tasks_id = fields.One2many(string ="Tasks",
32          comodel_name="manageleire.task",
33          inverse_name="sprint_id")
34      #Muchas sprint tienen un proyecto
35      project_id = fields.Many2one(comodel_name="manageleire.project",
36          string="project",
37          required=True,
38          ondelete="cascade")
39
40      #FUNCIONES
41      #función para calcular la fecha de fin (para duration)
42      #@api.depends define dependencias entre campos. Los métodos que usan este decorador se ejecutan automáticamente cuando cambian
43      @api.depends('start_date', 'duration')
44      def _get_end_date(self):
45          for sprint in self:
46              '''isinstance() se utiliza para verificar si un objeto pertenece a una clase.
47              Devuelve True si el objeto pertenece a la clase y False en caso contrario'''
48              if isinstance(sprint.start_date, datetime.datetime) and sprint.duration > 0:
49                  '''timedelta pertenece al módulo datetime y se utiliza para representar una
50                  diferencia o duración en tiempo (un intervalo de tiempo). Se usa para realizar operaciones con
51                  fechas: sumar o restar tiempo a una fecha, diferencia entre 2 fechas...
52                  Sintaxis: timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)
53                  Todos los argumentos son opcionales, se pueden marcar varios a la vez, y por defecto tienen el valor 0
54                  ...
55
56                  sprint.end_date = sprint.start_date + datetime.timedelta(days=sprint.duration)
57              else:
58                  sprint.end_date = sprint.start_date

```

- Task

```

models > task.py > task
1  # -*- coding: utf-8 -*-
2
3  import datetime
4  from odoo import models, fields, api # type: ignore
5
6  #se va a traducir en una tabla en la base de datos
7  class task(models.Model):
8
9      #ATRIBUTOS
10     _name = 'managelire.task' #nombreModulo.nombreModelo --> así le llamo desde Odoo
11     _description = 'managelire.task'
12
13
14      #CAMPOS
15      #entre paréntesis es el nombre de la tabla que se va a mostrar en la vista
16      name = fields.Char(string = "Nombre de la tarea", readonly = False, required = True, help = "Introduzca el nombre de la tarea")
17          #readonly (solo lectura): false --> se puede editar. Si fuese true no se podría editar
18          #required: True --> obligatorio
19          #cuando estoy sobre Nombre en la vista, se ve el mensaje de help
20      description = fields.Char(string = "Descripción")
21      start_date = fields.Datetime(string = "Fecha inicio")
22      end_date = fields.Datetime(string = "Fecha fin")
23      is_paused = fields.Boolean(string = "¿Está parado?")
24
25      #PRACTICA 17 - TEMA 8 --> Añadir al modelo tarea un nuevo campo computado llamado code, que no se almacene
26          #en la base de datos y que tome el siguiente valor: una cadena (TSK_, seguido del campo id del modelo)
27      code = fields.Char(string="Código", compute="_compute_code")
28
29      #TEMA 9 --> campo creado para almacenar la fecha y hora de creación de la tarea con una función lambda
30      definition_date = fields.Datetime(string = "Fecha de creación", default=lambda p: datetime.datetime.now())
31
32
33      #RELACIONES
34      #Cada sprint tiene múltiples tareas asignadas; cada tarea se asigna a un sprint específico
35      sprint_id = fields.Many2one(comodel_name="managelire.sprint",
36          string = "Sprint",
37          ondelete = "cascade",
38          #TEMA 9 --> vamos a tener un solo sprint abierto por proyecto en cada momento (fecha de inicio y de
39          #se van a asociar al sprint que esté abierto, de forma automática; el usuario no va a de
40          #compute=_get_sprint",
41          store= True)
42
43      #Cada tarea usa múltiples tecnologías y cada tecnología está asociada a múltiples tareas
44      tecnologias_id = fields.Many2many(string="Tecnologías",
45          comodel_name="managelire.technology", #es el obligatorio por defecto, es el nombre del modelo
46          relation="tecnologias_tareas", #nombre de la tabla que crea --> SE CONSULTA EN OODOO-AJUSTES-TE
47          column1="tecnologias_ids", #hace referencia al registro de la tabla actual (columna izquierda)
48          column2="tareas_ids") #hace referencia al registro de comodel_name (columna derecha)
49
50
51      #FUNCIONES
52      #PRACTICA 17 - TEMA 8 --> método del campo code
53      def _compute_code(self):
54          for task in self:
55              task.code = "TSK_"+ str(task.id) #id es un campo propio del programa de cada modelo
56
57
58      #TEMA 9 --> El campo sprint se va a calcular cuando se genere el valor del campo "code" (la función _get_sprint
59          #se va a ejecutar cuando el campo code cambie) por eso se crea la función _get_sprint, con el decorador @api.depends ("code").
60      @api.depends('code')
61      def _get_sprint(self):
62          for task in self: #recibe la colección de tareas
63              #obtener todos los sprints que tenemos en el sistema
64                  #self.env --> se introduce el modelo a buscar y sobre este modelo, queremos buscar los sprints asociados al proyecto en concreto; puede haber varios.
65                  #Para obtener los registros del modelo sprint: usamos search. Cada sprint tiene un campo project_id
66                  #relación Many2one; buscamos los sprints cuyo project_id sea igual al project_id.id de la historia de la
67                  #tarea. El resultado de esta búsqueda será un conjunto de registros (recordset) que cumplen con la condición.
68                  sprints = self.env['managelire.sprint'].search([('project_id', '=', task.history_id.project_id.id)])
69                  found = False
70
71                  #recorre la lista de sprints asociados al proyecto al que pertenece la tarea
72                  for sprint in sprints:
73                      #verificar que el campo end_date contiene una fecha válida (comprobar que es un objeto de tipo datetime: isinstance(sprint.end_date, datetime.datetime))
74                      #También comprobar si la fecha de finalización del sprint es posterior a la fecha actual
75                      if isinstance(sprint.end_date, datetime.datetime) and sprint.end_date > datetime.datetime.now():
76                          #si se encuentra un sprint válido, se asigna su ID al campo task.sprint_id y found pasa a valer True
77                          task.sprint_id = sprint
78                          found = True
79
80                  if not found:
81                      task.sprint_id = False
82
83
84      #AMPLIACIÓN: botón de eliminar que elimina el registro que se quiera. Se mostrará en el tree
85      def f_delete(self):
86          #eliminar el registro actual --> cuando se define un botón con type="object", el botón pasa automáticamente el registro actual (o los registros seleccionados) al método
87          if not self:
88              raise ValueError("No se seleccionó ningún registro para eliminar.")
89
90          #eliminar la tarea
91          self.unlink()

```

- **Technology**

```

models > ⌂ technology.py > ↗ technology > ↗ tareas_id
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api # type: ignore
4
5  #se va a traducir en una tabla en la base de datos
6  class technology(models.Model):
7
8      #ATRIBUTOS
9      _name = 'manageleire.technology' #nombreModulo.nombreModelo --> así le llamo desde Odoo
10     _description = 'manageleire.technology'
11
12
13     #CAMPOS
14     #entre paréntesis es el nombre de la tabla que se va a mostrar en la vista
15     name = fields.Char(string = "Nombre de la tecnología", readonly = False, required = True, help = "Introduzca el nombre de la tecnología")
16             #readonly (solo lectura): false --> se puede editar. Si fuese true no se podría editar
17             #required: True --> obligatorio
18             #cuando estoy sobre Nombre en la vista, se ve el mensaje de help
19     description = fields.Char(string = "Descripción")
20     photo = fields.Image(string = "Imagen")
21
22
23     #RELACIONES
24     #Cada tarea usa múltiples tecnologías y cada tecnología está asociada a múltiples tareas
25     tareas_id = fields.Many2many(string="tareas",
26             comodel_name = "manageleire.task", #modelo con el que se crea la relación
27             relation = "tasks_technologys", #nombre de la tabla que crea --> SE CONSULTA EN OODOO-AJUSTES-TECNICO-RELACION
28             column1 = "technologys_ids", #hace referencia al registro de la tabla actual (columna izquierda)
29             column2 = "tasks_ids") #hace referencia al registro de comodel_name (columna derecha)

```

3.3.3.2 SECURITY

```

security > ⌂ ir.model.access.csv
1  id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2
3  access_manageleire_task,manageleire.task,model_manageleire_task,base.group_user,1,1,1,1
4  access_manageleire_sprint,manageleire.sprint,model_manageleire_sprint,base.group_user,1,1,1,1
5  access_manageleire_project,manageleire.project,model_manageleire_project,base.group_user,1,1,1,1
6  access_manageleire_history,manageleire.history,model_manageleire_history,base.group_user,1,1,1,1
7  access_manageleire_technology,manageleire.technology,model_manageleire_technology,base.group_user,1,1,1,1

```

3.3.3.3 VIEWS

- **Developer**

```

views > developer.xml
 1  <!--hacer visible la relación M2M de technologies desde el menú del módulo, para ver qué vistas toma por defecto:
 2  tomará las vistas de res.partner. Crear solamente el manage.action_developer_window y una opción de menú: menuitem para Developers-->
 3  <odoo>
 4    <data>
 5      <!--FORM-->
 6      <record model="ir.ui.view" id="manageleire.devs_partner_form">
 7        <field name="name">manageleire.devs_partner_form</field>
 8        <field name="model">res.partner</field>
 9        <field name="inherit_id" ref="base.view_partner_form"></field>
10        <field name="mode">primary</field>
11        <field name="arch" type="xml">
12          <xpath expr="//sheet/notebook/page[@name='internal_notes']" position="after">
13            <page name="devs" string="Devs">
14              <group>
15                <group>
16                  <field name="technologies"/>
17                  <field name="is_dev"/>
18                </group>
19              </group>
20            </page>
21          </xpath>
22        </field>
23      </record>
24
25
26
27      <!--ACTIONS-->
28      <record model="ir.actions.act_window" id="manageleire.action_developer_window">
29        <field name="name">Listado de desarrolladores</field>
30        <field name="res_model">res.partner</field>
31        <field name="view_mode">tree,form</field>
32        <field name="domain">[('is_dev', '!=', False)]</field>
33      </record>
34
35      <record model="ir.actions.act_window.view" id="manageleire.action_view_developer_tree">
36        <field name="sequence" eval="1"/>
37        <field name="view_mode">tree</field>
38        <field name="view_id" ref="base.view_partner_tree"/>
39        <field name="act_window_id" ref="manageleire.action_developer_window"></field>
40      </record>
41
42      <record model="ir.actions.act_window.view" id="manageleire.action_view_developer_form">
43        <field name="sequence" eval="2"/>
44        <field name="view_mode">form</field>
45        <field name="view_id" ref="manageleire.devs_partner_form"/>
46        <field name="act_window_id" ref="manageleire.action_developer_window"/>
47      </record>
48
49      <!-- menú raíz -->
50      <menuitem name= "Manage de Leire" id= "menu_manageleire_raiz"/> <!--El nombre es lo que se va a
51
52      <!--Segundo nivel-->
53      <menuitem name="MANAGE" id= "menu_manageleire_manage" parent= "menu_manageleire_raiz"/>
54
55      <!--Tercer nivel donde se lleva a cabo -> actions-->
56      <menuitem name="Developers" id= "menu_manageleire_developer" parent= "menu_manageleire_manage"
57        | action="manageleire.action_developer_window"/> <!--action: id de la plantilla de action-->
58    </data>
59  </odoo>

```

- History

```

views > history.xml
1  <odoo>
2    <data>
3
4      <!-- el tree -> saca los datos en una tabla-->
5      <record model="ir.ui.view" id="vista_manageleire_history_tree">
6        <field name="name">vista_manageleire_history_tree</field>
7        <field name="model">manageleire.history</field>
8        <field name="arch" type="xml">
9          <tree>
10            <!-- le indico los campos (de history.py) que quiero que se muestren en la tabla -->
11            <field name= "name"/>
12            <field name= "description"/>
13        </tree>
14      </field>
15    </record>
16
17
18      <!-- Plantilla formulario tipo form -> el formulario que sale cuando voy a añadir un nuevo history-->
19      <record model="ir.ui.view" id="vista_manageleire_history_form"> <!--form pq se trata de un formulario-->
20        <field name="name">vista_manageleire_history_form</field>
21        <field name="model">manageleire.history</field>
22        <field name="arch" type="xml">
23          <form string="formulario_history" >
24            <sheet>
25              <group name="group_top">
26                <group name="group_left">
27                  <field name="name"/>
28                  <field name="description"/>
29                </group>
30
31                <group name="group_right">
32                  <field name="project_id"/>
33                </group>
34              </group>
35            </sheet>
36          </form>
37        </field>
38      </record>
39
40      <!-- Plantilla action -->
41      <record model="ir.actions.act_window" id="accion_manageleire_history_form">
42        <field name="name">Listado de history</field>
43        <field name="type">ir.actions.act_window</field>
44        <field name="res_model">manageleire.history</field>
45        <field name="view_mode">tree,form</field>
46        <field name="help" type="html">
47          <p class="oe_view_nocontent_create">
48            History
49          </p>
50          <p> Click <strong> 'Crear' </strong> para añadir nuevos elementos
51          </p>
52        </field>
53      </record>
54
55
56
57
58      <!-- menú raíz -->
59      <menuitem name= "Manage de Leire" id= "menu_manageleire_raiz"/> <!--El nombre es lo que se va
60
61      <!--Segundo nivel-->
62      <menuitem name="MANAGE" id= "menu_manageleire_manage" parent= "menu_manageleire_raiz"/>
63
64      <!--Tercer nivel donde se lleva a cabo -> actions-->
65      <menuitem name="History" id= "menu_manageleire_history" parent= "menu_manageleire_manage"
66        action="accion_manageleire_history_form"/> <!--action: id de la plantilla de action-->
67
68    </data>
69  </odoo>

```

- Project

```

views > project.xml
1  <odoo>
2    <data>
3
4      <!--el tree -> saca los datos en una tabla-->
5      <record model="ir.ui.view" id="vista_manageleire_project_tree">
6        <field name="name">vista_manageleire_project_tree</field>
7        <field name="model">manageleire.project</field>
8        <field name="arch" type="xml">
9          <tree>
10            <!--le indico los campos (de project.py) que quiero que se muestren en la tabla -->
11            <field name= "name"/>
12            <field name= "description"/>
13          </tree>
14        </field>
15      </record>
16
17
18      <!-- Plantilla formulario tipo form -> el formulario que sale cuando voy a añadir un nuevo project-->
19      <record model="ir.ui.view" id="vista_manageleire_project_form"> <!--form pq se trata de un formulario-->
20        <field name="name">vista_manageleire_project_form</field>
21        <field name="model">manageleire.project</field>
22        <field name="arch" type="xml">
23          <form string="formulario_project" >
24            <sheet>
25              <group name="group_top">
26                <field name="name"/>
27                <field name="description"/>
28              </group>
29            </sheet>
30          </form>
31        </field>
32      </record>
33
34
35      <!-- Plantilla action -->
36      <record model="ir.actions.act_window" id="accion_manageleire_project_form">
37        <field name="name">Listado de project</field>
38        <field name="type">ir.actions.act_window</field>
39        <field name="res_model">manageleire.project</field>
40        <field name="view_mode">tree,form</field>
41        <field name="help" type="html">
42          <p class="oe_view_nocontent_create">
43            Project
44          </p>
45          <p> Click <strong> 'Crear' </strong> para añadir nuevos elementos
46          </p>
47        </field>
48      </record>
49
50      <!-- menú raíz -->
51      <menuitem name= "Manage de Leire" id= "menu_manageleire_raiz"/> <!--El nombre es lo que se v
52
53      <!--Segundo nivel-->
54      <menuitem name="MANAGE" id= "menu_manageleire_manage" parent= "menu_manageleire_raiz"/>
55
56      <!--Tercer nivel donde se lleva a cabo -> actions-->
57      <menuitem name="Project" id= "menu_manageleire_project" parent= "menu_manageleire_manage"
58        action="accion_manageleire_project_form"/> <!--action: id de la plantilla de action-->
59
60      </data>
61  </odoo>
62

```

- Sprint

```
views > sprint.xml
1  <ooodoo>
2    <data>
3
4      <!--el tree -> saca los datos en una tabla-->
5      <record model="ir.ui.view" id="vista_manageleire_sprint_tree">
6        <field name="name">vista_manageleire_sprint_tree</field>
7        <field name="model">manageleire.sprint</field>
8        <field name="arch" type="xml">
9          <tree>
10            <!--le indico los campos (de sprint.py) que quiero que se muestren en la tabla -->
11            <field name= "name"/>
12            <field name= "description"/>
13            <field name= "start_date"/>
14            <field name = "end_date"/>
15          </tree>
16        </field>
17      </record>
18
19
20      <!-- Plantilla formulario tipo form -> el formulario que sale cuando voy a añadir un nuevo sprint-->
21      <record model="ir.ui.view" id="vista_manageleire_sprint_form"> <!--form pq se trata de un formulario-->
22        <field name="name">vista_manageleire_sprint_form</field>
23        <field name="model">manageleire.sprint</field>
24        <field name="arch" type="xml">
25          <form string="formulario_sprint" >
26            <sheet>
27              <group name="group_top">
28                <group name="group_left">
29                  <field name="name"/>
30                  <field name="description"/>
31                  <field name="start_date"/>
32                  <field name="end_date"/>
33                  <field name="duration"/>
34                </group>
35
36                <group name="group_right">
37                  <field name="project_id"/>
38                </group>
39              </group>
40            </sheet>
41          </form>
42        </field>
43      </record>
44
45      <!-- Plantilla action -->
46      <record model="ir.actions.act_window" id="accion_manageleire_sprint_form">
47        <field name="name">Listado de sprint</field>
48        <field name="type">ir.actions.act_window</field>
49        <field name="res_model">manageleire.sprint</field>
50        <field name="view_mode">tree,form</field>
51        <field name="help" type="html">
52          <p class="oe_view_nocontent_create">
53            Sprint
54          </p>
55          <p> Click <strong> 'Crear' </strong> para añadir nuevos elementos
56          </p>
57        </field>
58      </record>
59
60
61
62
63      <!-- menú raíz -->
64      <menuitem name= "Manage de Leire" id= "menu_manageleire_raiz"/> <!--El nombre es lo que se
65
66      <!--Segundo nivel-->
67      <menuitem name="MANAGE" id= "menu_manageleire_manage" parent= "menu_manageleire_raiz"/>
68
69      <!--Tercer nivel donde se lleva a cabo -> actions-->
70      <menuitem name="Sprint" id= "menu_manageleire_sprint" parent= "menu_manageleire_manage"
71        | action="accion_manageleire_sprint_form"/> <!--action: id de la plantilla de action-->
72    </data>
73  </ooodoo>
```

- Task

```

views > task.xml
1  <ooodo>
2    <data>
3
4      <!--el tree -> saca los datos en una tabla-->
5      <record model="ir.ui.view" id="vista_manageleire_task_tree">
6        <field name="name">vista_manageleire_task_tree</field>
7        <field name="model">manageleire.task</field>
8        <field name="arch" type="xml">
9          <tree>
10            <!--le indico los campos (de task.py) que quiero que se muestren en la tabla -->
11            <field name= "name"/>
12            <field name= "description"/>
13            <field name="definition_date"/>
14            <field name= "start_date"/>
15            <field name = "end_date"/>
16            <field name = "is_paused"/>
17            <!--Botón de eliminar -> ampliación-->
18            <button name = "f_delete" string="Eliminar" class="oe_hightlight" type="object"/>
19          </tree>
20        </field>
21      </record>
22
23      <!-- Plantilla formulario tipo form -> el formulario que sale cuando voy a añadir un nuevo task-->
24      <record model="ir.ui.view" id="vista_manageleire_task_form" > <!--form pq se trata de un formulario-->
25        <field name="name">vista_manageleire_task_form</field>
26        <field name="model">manageleire.task</field>
27        <field name="arch" type="xml">
28          <form string="formulario_task" >
29            <sheet>
30              <group name="group_top">
31                <!--Formulario con vista de dos columnas-->
32                <group name="group_left">
33                  <field name="code"/>
34                  <field name="definition_date"/>
35                  <field name="name"/>
36                  <field name="description"/>
37                  <field name="start_date"/>
38                  <field name="end_date"/>
39                </group>
40
41                <group name="group_right">
42                  <field name="is_paused"/>
43                  <!--visualizo las relaciones-->
44                  <field name="tecnologias_id"/>
45                  <field name="sprint_id"/>
46                  <field name="history_id"/>
47                </group>
48              </group>
49            </sheet>
50          </form>
51        </field>
52      </record>
53
54      <!-- Plantilla action -->
55      <record model="ir.actions.act_window" id="accion_manageleire_task_form">
56        <field name="name">Listado de tareas</field>
57        <field name="type">ir.actions.act_window</field>
58        <field name="res_model">manageleire.task</field>
59        <field name="view_mode">tree,form</field>
60        <field name="help" type="html">
61          <p class="oe_view_nocontent_create">
62            Task
63          </p>
64          <p> Click <strong> 'Crear' </strong> para añadir nuevos elementos
65          </p>
66        </field>
67      </record>
68
69
70
71
72      <!-- menú raíz -->
73      <menuitem name= "Manage de Leire" id= "menu_manageleire_raiz"/> <!--El nombre es lo que se
74
75      <!--Segundo nivel-->
76      <menuitem name="MANAGE" id= "menu_manageleire_manage" parent= "menu_manageleire_raiz"/>
77
78      <!--Tercer nivel donde se lleva a cabo -> actions-->
79      <menuitem name="Task" id= "menu_manageleire_task" parent= "menu_manageleire_manage"
80        action="accion_manageleire_task_form"/> <!--action: id de la plantilla de action-->
81
82    </data>
83  </ooodo>

```

- Technology

```

views > technology.xml
1  <odoo>
2    <data>
3
4      <!-- el tree -> saca los datos en una tabla-->
5      <record model="ir.ui.view" id="vista_manageleire_technology_tree">
6        <field name="name">vista_manageleire_technology_tree</field>
7        <field name="model">manageleire.technology</field>
8        <field name="arch" type="xml">
9          <tree>
10            <!-- le indico los campos (de technology.py) que quiero que se muestren en la tabla -->
11            <field name= "name"/>
12            <field name= "description"/>
13            <field name= "photo"/>
14          </tree>
15        </field>
16      </record>
17
18
19      <!-- Plantilla formulario tipo form -> el formulario que sale cuando voy a añadir un nuevo technology-->
20      <record model="ir.ui.view" id="vista_manageleire_technology_form" <!--form pq se trata de un formulario-->
21        <field name="name">vista_manageleire_technology_form</field>
22        <field name="model">manageleire.technology</field>
23        <field name="arch" type="xml">
24          <form string="formulario_technology" >
25            <sheet>
26              <group name="group_top">
27                <!--Formulario con vista de dos columnas-->
28                <group name="group_left">
29                  <field name="name"/>
30                  <field name="description"/>
31                </group>
32
33                <group name="group_right">
34                  <field name="photo" widget="image"/>
35                  <!--visualizo la relación M2M-->
36                  <field name="tareas_id"/>
37                </group>
38              </group>
39            </sheet>
40          </form>
41        </field>
42      </record>
43
44      <!--Modelo Kanban: dar formato de tarjeta a las tecnologías-->
45      <record model="ir.ui.view" id="vista_manageleire_technology_kanban">
46        <field name="name">vista_manageleire_technology_kanban</field>
47        <field name="model">manageleire.technology</field>
48        <field name="arch" type="xml">
49          <kanban>
50            <!--los campos que quiero que se muestren en la vista. poner siempre el id aunque no se vaya a mostrar-->
51            <field name="id"/>
52            <field name= "name"/>
53            <field name="photo"/>
54            <templates>
55              <t t-name="kanban_box" <!--definir una plantilla Qweb que anida al resto de elementos-->
56              <div t-attf-class="oe_kanban_global_click">
57                <!--dividido en 2 partes (en una aparece la imagen y en la otra aparece el texto con los detalles del registro)-->
58                <div class="oe_kanban_image">
59                  <!--En el primer parámetro de la función hay que poner el nombre del modelo que sirve para que se puedan interpretar los datos del registro en crudo, y en el segundo, el nombre del campo del modelo que contiene la imagen.-->
60                   <!--el primer parámetro es el nombre del modelo y el segundo es el nombre del campo-->
61                </div>
62                <div class="oe_kanban_details">
63                  <!--en este punto, indico los campos que quiero mostrar de los ya descritos arriba-->
64                  <strong class="oe_kanban_record_title" <!--_oe_kanban_record_title: muestra los textos resaltados, como títulos-->
65                    <field name="name"/> <!--name es el nombre de la tecnología-->
66                  </strong>
67                </div>
68              </t>
69            </templates>
70          </kanban>
71        </field>
72      </record>
73
74
75
76
77
78

```

```

81 <!-- Plantilla action -->
82 <record model="ir.actions.act_window" id="accion_manageleire_technology_form">
83   <field name="name">Listado de tecnologías</field>
84   <field name="type">ir.actions.act_window</field>
85   <field name="res_model">manageleire.technology</field>
86   <field name="view_mode">tree,form,kanban</field> <!--añadir para que se vea el modelo kanban-
87   <field name="help" type="html">
88     <p class="oe_view_nocontent_create">
89       | Technology
90     </p>
91     <p> Click <strong> 'Crear' </strong> para añadir nuevos elementos
92     </p>
93   </field>
94 </record>

95
96
97 <!-- menú raiz -->
98 <menuitem name= "Manage de Leire" id= "menu_manageleire_raiz"/> <!--El nombre es lo que se va a
99
100 <!--Segundo nivel-->
101 <menuitem name="MANAGE" id= "menu_manageleire_manage" parent= "menu_manageleire_raiz"/>
102
103 <!--Tercer nivel donde se lleva a cabo -> actions-->
104 <menuitem name="Technology" id= "menu_manageleire_technology" parent= "menu_manageleire_manage"
105   | action="accion_manageleire_technology_form"/> <!--action: id de la plantilla de action-->
106 </data>
107 </odoor>

```

3.3.3.4 __init__.py

```

__init__.py x
__init__.py
1  # -*- coding: utf-8 -*-
2
3  from . import controllers
4  from . import models

```

3.3.3.5 MANIFEST

```

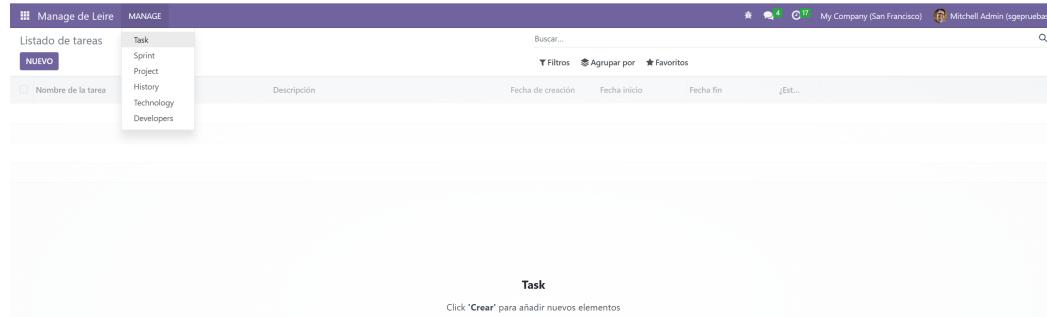
__manifest__.py
1  # -*- coding: utf-8 -*-
2  {
3    'name': "manageleire",
4
5    'summary': """
6      Short (1 phrase/line) summary of the module's purpose, used as
7      subtitle on modules listing or apps.openerp.com""",
8
9    'description': """
10      Long description of module's purpose
11      """,
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', #quitar de estar comentado para que salga lo de manage de Leire
28      'views/views.xml',
29      'views/task.xml',
30      'views/sprint.xml',
31      'views/project.xml',
32      'views/history.xml',
33      'views/technology.xml',
34      'views/developer.xml',
35      'views/templates.xml',
36    ],
37    # only loaded in demonstration mode
38    'demo': [
39      'demo/demo.xml',
40    ],
41  }

```

4 PRUEBAS DE FUNCIONAMIENTO

- Visión inicial de Odoo del “managelire”

En la imagen adjunta se muestra la estructura final de mi proyecto, organizado de manera que todo parte de un único apartado principal, desde el cual se desglosan los demás.



The screenshot shows the Odoo interface for the 'Manage Leire' project. The top navigation bar includes 'Manage Leire' and 'MANAGE'. Below it, a sidebar lists 'Listado de tareas' and a 'NUEVO' button. A dropdown menu under 'NUEVO' shows options: Task, Sprint, Project, History, Technology, and Developers. The main content area is titled 'Task' and contains a table with columns: Descripción, Fecha de creación, Fecha inicio, Fecha fin, and '¿Está parado?'. The table lists six tasks: Tarea 1 through Tarea 6. At the bottom of the table, there is a note: 'Click "Crear" para añadir nuevos elementos'.

- Task

Visión del apartado Task, tras la creación de diferentes tareas.

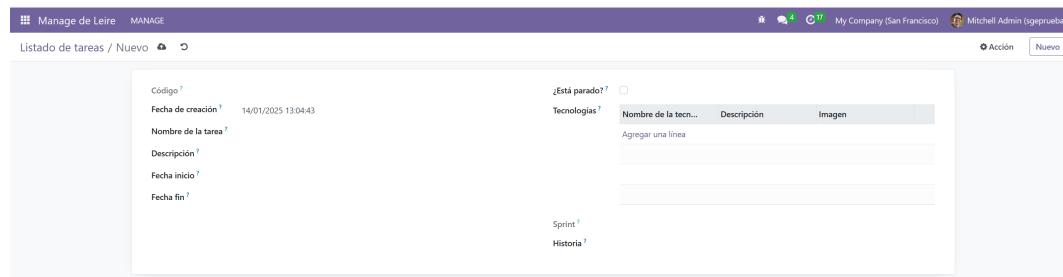


This screenshot shows the 'Task' list after creating six tasks. The table has columns: Nombre de la tarea, Descripción, Fecha de creación, Fecha inicio, Fecha fin, and ¿Está parado?. The tasks are: Tarea 1 (14/01/2025 12:58:30), Tarea 2 (14/01/2025 13:00:57), Tarea 3 (14/01/2025 13:02:45), Tarea 4 (14/01/2025 13:03:08), Tarea 5 (14/01/2025 13:03:25), and Tarea 6 (14/01/2025 13:03:45). Each task has a checkbox next to its name and a 'Eliminar' (Delete) link to its right. The page footer indicates '1-6 / 6'.

- Formulario Task

En la siguiente imagen se puede ver cómo es el formulario para crear una nueva tarea. Cada tarea usa ciertas tecnologías y está diseñada para realizar una historia. Hay diferentes campos para llenar, pero cabe destacar, los dos campos que se autocompletan tras la creación:

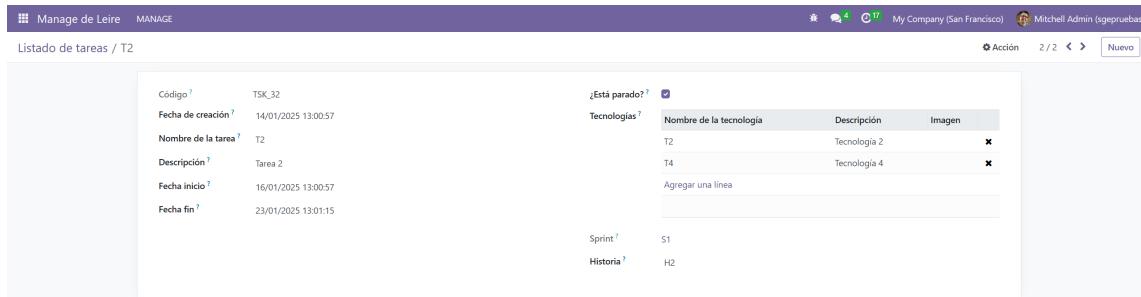
- Campo código: se genera automáticamente con un formato del tipo “TSK_id”, donde id es un valor autoincremental asignado a cada nueva tarea creada. Este comportamiento está definido en un método específico.
- Campo fecha de creación: de manera automática se completa con la fecha y hora de creación, pero puede modificarse si se desea.
- Campo sprint: se asigna automáticamente si la fecha de inicio coincide con el inicio de un sprint o si se encuentra dentro del rango de fechas de un sprint en curso. En caso de que no ocurra, se quedará vacío el campo.



This screenshot shows the 'Nuevo' (New) task creation form. It includes fields for Código (auto-generated), Fecha de creación (14/01/2025 13:04:43), Nombre de la tarea, Descripción, Fecha inicio, Fecha fin, Tecnologías (checkbox), Nombre de la tecn., Descripción, and Imagen. There is also a section for Sprint and Historia. A note at the bottom says 'Agregar una línea'.

- **Ejemplo formulario task**

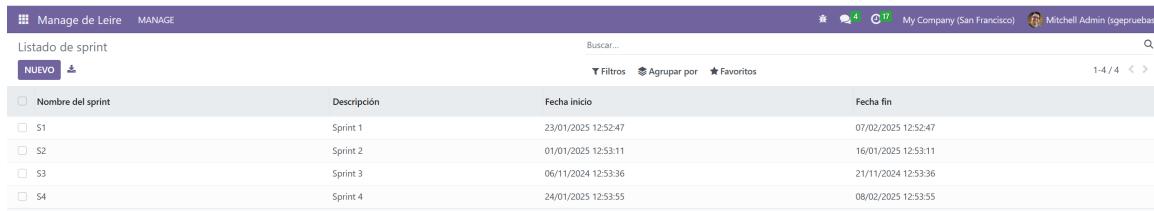
Se puede ver un ejemplo práctico de como quedaría uno de los formularios creados para una tarea.



Nombre de la tecnología	Descripción	Imagen
T2	Tecnología 2	X
T4	Tecnología 4	X
Agregar una línea		

- **Sprint**

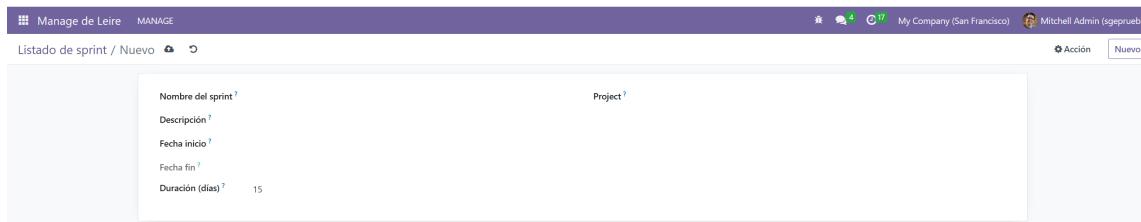
Visión del apartado Sprint, tras la creación de diferentes sprints.



Nombre del sprint	Descripción	Fecha inicio	Fecha fin
S1	Sprint 1	23/01/2025 12:52:47	07/02/2025 12:52:47
S2	Sprint 2	01/01/2025 12:53:11	16/01/2025 12:53:11
S3	Sprint 3	06/11/2024 12:53:36	21/11/2024 12:53:36
S4	Sprint 4	24/01/2025 12:53:55	08/02/2025 12:53:55

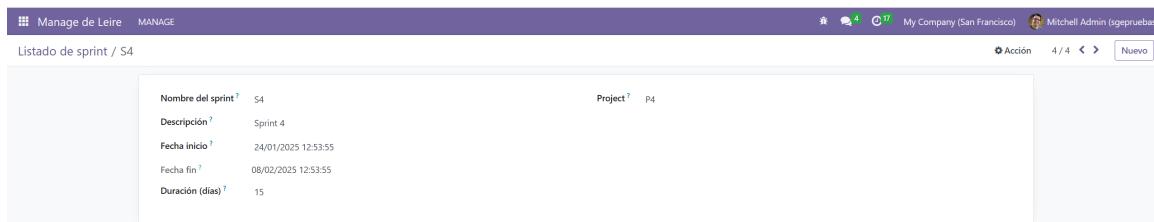
- **Formulario Sprint**

Este es el formulario que se debe llenar para crear un nuevo sprint. Llama la atención que la fecha de fin no puede modificarse. Esto es debido a que depende de la fecha de inicio y a la duración, en días. Funciona de la siguiente manera: se marca una duración, en días, y se escoge una fecha de inicio, y, automáticamente, la fecha de fin se completa. Cada sprint está asociado a un proyecto.



- **Ejemplo formulario sprint**

Este es un ejemplo de un formulario relleno de un sprint. Es importante fijarse en que la fecha de fin ha sido completada en función de la fecha de inicio y de la duración.



- **Project**

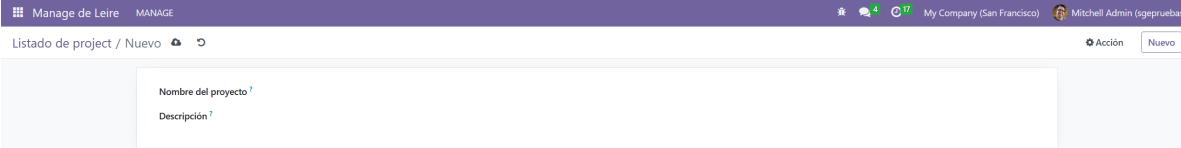
Visión del apartado Project, tras la creación de diferentes proyectos.



Nombre del proyecto	Descripción
P1	Proyecto 1
P2	Proyecto 2
P3	Proyecto 3
P4	Proyecto 4

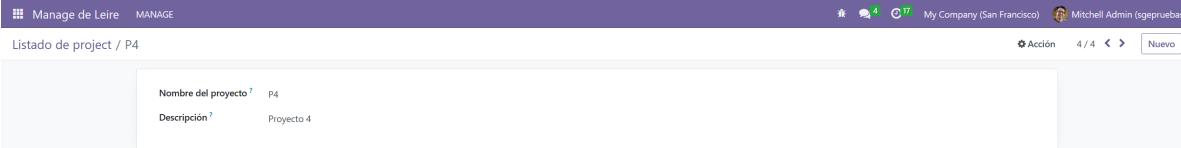
- **Formulario Project**

Este es el formulario que se debe llenar para crear un proyecto.



- **Ejemplo formulario Project**

Aquí, se muestra un ejemplo de la creación de un nuevo proyecto.



- **History**

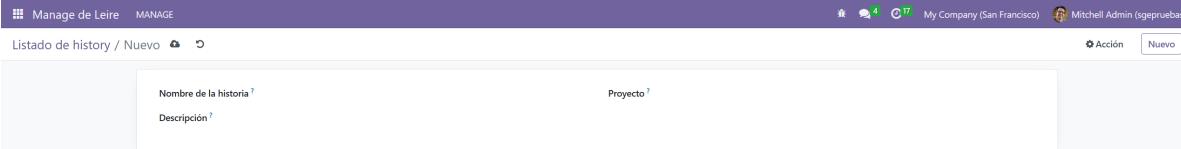
Visión del apartado History, tras la creación de diferentes historias.



Nombre de la historia	Descripción
H1	Historia 1
H2	Historia 2
H3	Historia 3
H4	Historia 4

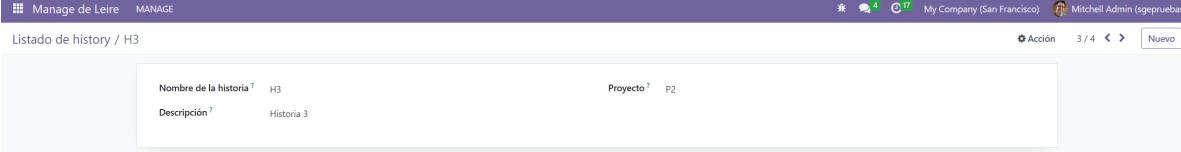
- **Formulario History**

Este es el formulario que hay que rellenar para crear una historia, la cual depende de un proyecto.



- **Ejemplo formulario History**

Un ejemplo de la creación de una historia.



- **Technology**

Visión del apartado Technology, tras la creación de diferentes tecnologías.



Nombre de la tecnología	Descripción	Imagen
T1	Tecnología 1	103.74 Kb
T2	Tecnología 2	8.20 Kb
T3	Tecnología 3	9.54 Kb
T4	Tecnología 4	4.79 Kb

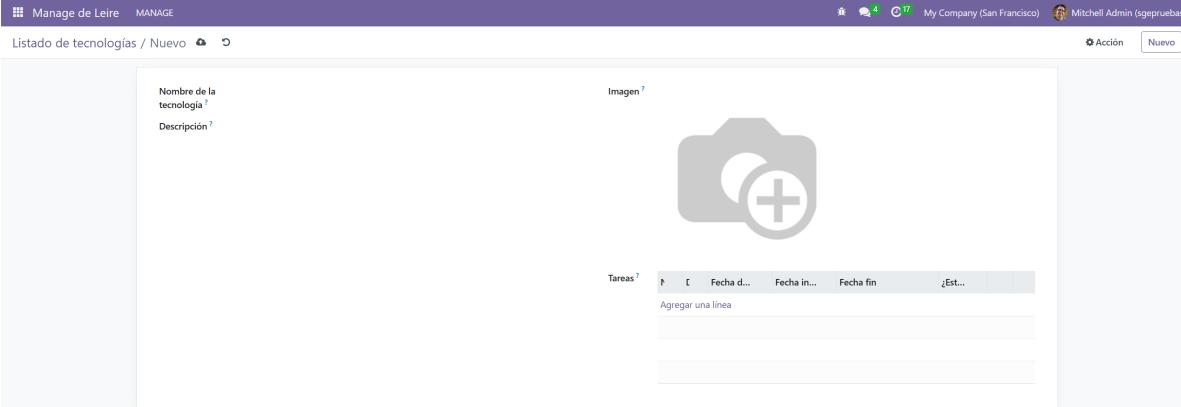
- **Technology en formato Kanban**

Aquí se ve una de las ampliaciones que he realizado, que es la implementación del modelo Kanban.



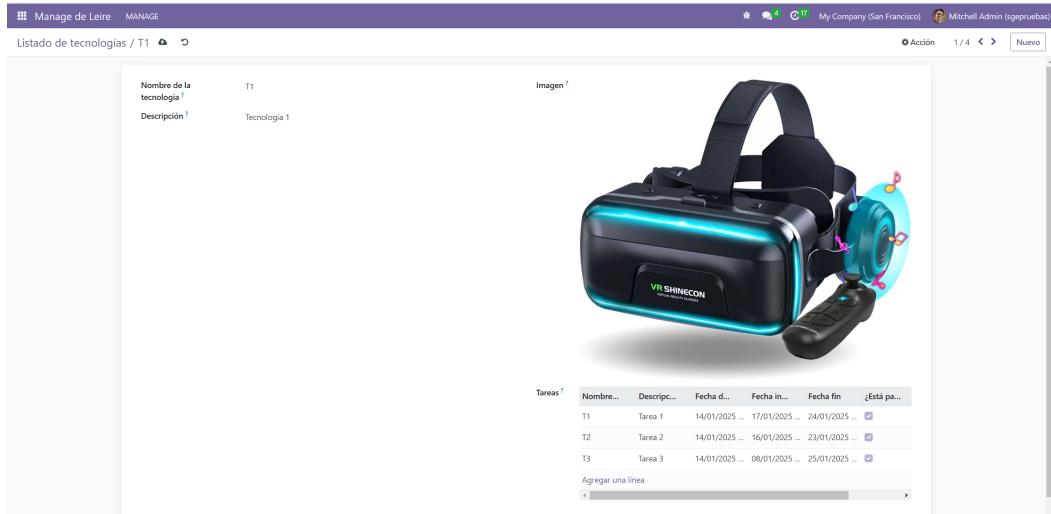
- **Formulario Technology**

En la imagen se muestra el formulario diseñado para añadir una nueva tecnología. Esta tecnología puede ser utilizada en diversas tareas, lo que justifica su representación en formato tree. Además, el formulario incluye la opción de adjuntar una foto.



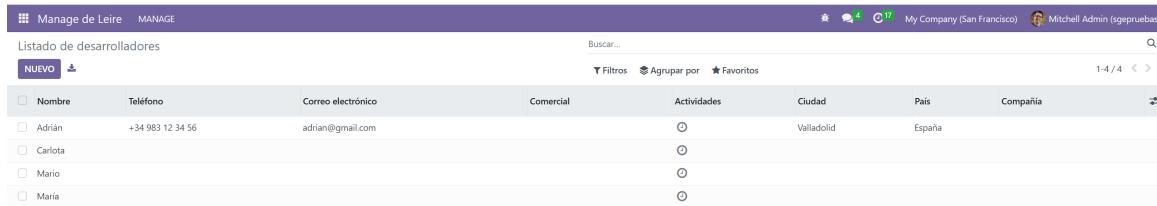
- **Ejemplo formulario technology**

Ejemplo del formulario de una nueva tecnología.



- **Developer**

La siguiente imagen muestra desarrolladores creados.

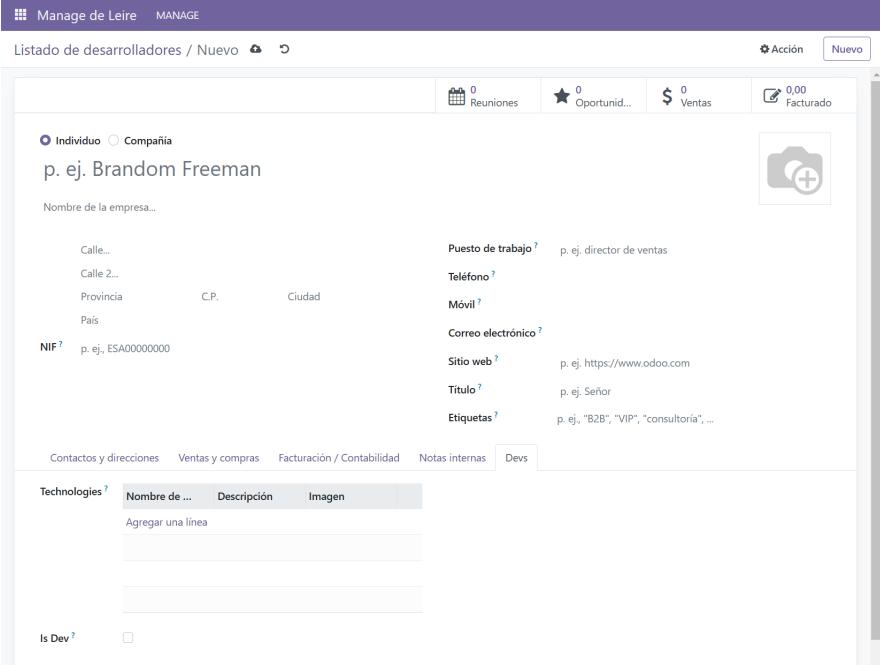


The screenshot shows a Odoo module interface titled "Manage de Leire" under the "MANAGE" tab. It displays a list of developers. A new developer, "Adrián", has been added and is listed at the top. The table includes columns for Nombre, Teléfono, Correo electrónico, Comercial, Actividades, Ciudad, País, and Compañía. A "NUEVO" button is visible at the top left.

<input type="checkbox"/> Nombre	Teléfono	Correo electrónico	Comercial	Actividades	Ciudad	País	Compañía
<input type="checkbox"/> Adrián	+34 983 12 34 56	adrian@gmail.com	<input type="radio"/>	<input type="radio"/>	Valladolid	España	
<input type="checkbox"/> Carlota			<input type="radio"/>	<input type="radio"/>			
<input type="checkbox"/> Mario			<input type="radio"/>	<input type="radio"/>			
<input type="checkbox"/> María			<input type="radio"/>	<input type="radio"/>			

- **Formulario Developer**

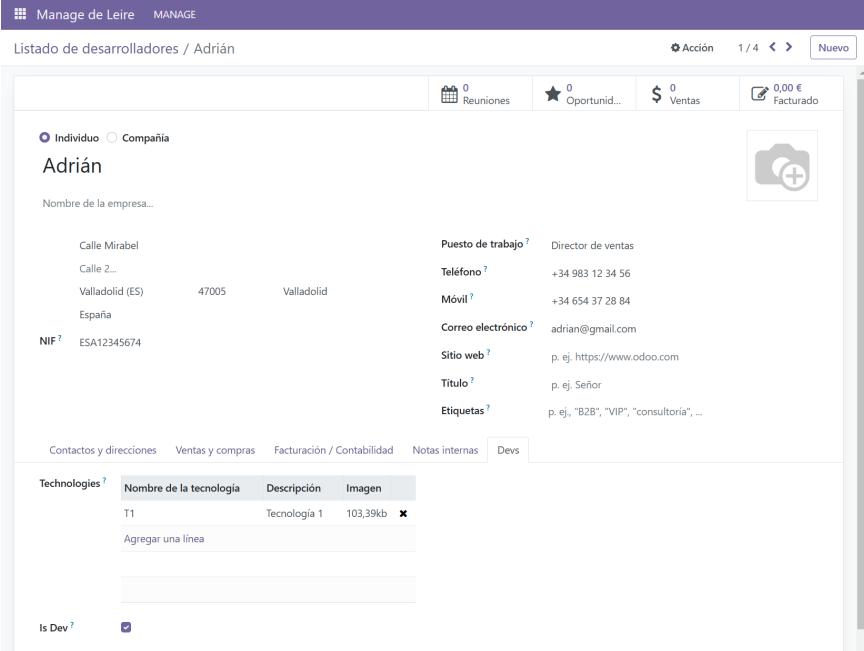
En la siguiente imagen se presenta el formulario utilizado para añadir un nuevo individuo o compañía. Se observa la relación Many2many con Technologies de tipo tree, lo que permite relacionar al desarrollador con diferentes tecnologías. Es importante destacar que, para almacenar correctamente el registro, es necesario activar la casilla "Is dev", la cual indica si la persona es desarrolladora o no.



The screenshot shows the Odoo 'Manage de Leire' interface for creating a new developer. At the top, there are tabs for 'Manage' and 'Nuevo'. Below the tabs, there are summary statistics: 0 Reuniones, 0 Oportunid..., \$ 0 Ventas, and 0,00 Facturado. A large 'Nuevo' button is visible. The main form has two radio buttons: 'Individuo' (selected) and 'Compañía'. A placeholder text 'p. ej. Brandon Freeman' is present. A file upload field with a camera icon and a plus sign is shown. Below the form fields are sections for address (Calle, Calle 2, Provincia, C.P., Ciudad), identification (NIF), and contact information (Puesto de trabajo, Teléfono, Móvil, Correo electrónico, Sitio web, Título, Etiquetas). A 'Technologies' section contains a table with columns 'Nombre de la tecnología', 'Descripción', and 'Imagen'. A row for 'T1' is listed with 'Tecnología 1' and '103,39kb'. At the bottom, there is a checkbox for 'Is Dev'.

- **Ejemplo formulario developer**

Ejemplo del formulario de uno de los desarrolladores.



This screenshot shows the Odoo 'Manage de Leire' interface for the developer 'Adrián'. The top bar includes 'Manage', 'Nuevo', and navigation icons. Summary statistics show 0 Reuniones, 0 Oportunid..., \$ 0 Ventas, and 0,00 € Facturado. The 'Nuevo' button is visible. The form displays personal information: 'Individuo' selected, 'Nombre de la empresa...' (Adrián), address (Calle Mirabel, Valladolid, Spain), NIF (ESA12345674), and contact details (Puesto de trabajo: Director de ventas, Teléfono: +34 983 12 34 56, Móvil: +34 654 37 28 84, Correo electrónico: adrian@gmail.com, Sitio web: https://www.odoo.com, Título: Señor). The 'Technologies' section shows a single entry: 'T1' with 'Tecnología 1' and '103,39kb'. The 'Is Dev' checkbox is checked.

5 AMPLIACIÓN

He realizado dos ampliaciones en el proyecto; la primera consiste en la incorporación de un botón específico para eliminar tareas, y la segunda consiste en la implementación de una vista en formato Kanban para las tecnologías.

5.1 BOTÓN DE ELIMINAR TAREAS

He añadido un botón específico para eliminar tareas directamente desde la interfaz, con el objetivo de mejorar la experiencia del usuario al simplificar esta acción, evitando pasos innecesarios o múltiples clics. Esto facilita la gestión de tareas y contribuye a una interacción más intuitiva con la aplicación, especialmente porque el botón está ubicado en la visión general, lo que permite eliminar tareas sin necesidad de entrar en cada una de ellas.

5.2 VISTA KANBAN

He implementado una vista en formato Kanban que organiza las tecnologías de manera visual, utilizando tarjetas que incluyen imágenes representativas. El objetivo principal es ofrecer una representación más clara y atractiva de las tecnologías, lo que facilita su identificación y mejora la experiencia visual del usuario.

6 CONCLUSIONES Y POSIBLES AMPLIACIONES

El proyecto en el que he estado trabajando ha sido fundamental para profundizar en el desarrollo de módulos en Odoo. A través de este proceso, he podido aplicar diversos conceptos clave, como la creación de campos computados y campos por defecto en la interfaz de usuario. Esto me ha permitido mejorar la funcionalidad del sistema y ofrecer una experiencia más personalizada a los usuarios, optimizando los procesos de gestión.

El uso de campos computados ha sido esencial para implementar funcionalidades que requieren el cálculo dinámico de valores, lo que ha mejorado la eficiencia en la gestión de datos. Por otro lado, los campos por defecto han permitido simplificar el flujo de trabajo, asegurando que los usuarios puedan completar formularios con valores predeterminados y evitando errores manuales.

Posibles ampliaciones

A pesar de las ampliaciones realizadas, aún hay varios aspectos que se pueden mejorar:

- Añadir modelo Kanban también en los proyectos: me parece que es una forma muy útil de obtener una visión clara y concreta de los proyectos. Para llevarlo a cabo, es necesario añadir un nuevo atributo, el de imagen, para poder implementarlo correctamente.
- Asignación de tareas o proyectos a diferentes desarrolladores: me refiero a que los desarrolladores tengan la opción de centrarse en un solo proyecto o en una tarea que los involucre en diferentes proyectos, pero de manera más específica. Sin embargo, también se debe permitir la posibilidad de no estar asignados a nada específico y poder ser asignados donde más se necesite, según la demanda.

7 ENLACE AL PROYECTO MANAGELEIRE

El siguiente enlace es una entrada a un repositorio donde he ido subiendo las entregas más pesadas. Debes seleccionar aquella cuyo título es “manageleire”.

https://github.com/leire-yagfer/SGE_ENTREGAS.git

8 BIBLIOGRAFÍA

- Oracle. (s. f.). ¿Qué es un ERP?: [https://www.oracle.com/es/erp/what-is-erp/#:~:text=Enterprise%20Resource%20Planning%20\(ERP\)%20es,de%20la%20cadena%20de%20suministro](https://www.oracle.com/es/erp/what-is-erp/#:~:text=Enterprise%20Resource%20Planning%20(ERP)%20es,de%20la%20cadena%20de%20suministro).
- Proyectos Ágiles. ¿Qué es Scrum?: <https://proyectosagiles.org/que-es-scrum/>
- Calipso. Historia del ERP: <https://www.calipso.com/articulos/historia-del-erp-origen-linea-de-tiempo-y-futuro/>
- Nanobytes. Odoo: <https://nanobytes.es/odoo#:~:text=Es%20una%20herramienta%20que%20permite,tanto%20la%20pérdida%20de%20datos>
- Ionos. Tutorial Docker: instalación y primeros pasos: <https://www.ionos.es/digitalguide/servidores/configuracion/tutorial-docker-instalacion-y-primeros-pasos/>
- Docker. Apuntes tema 4 del Moodle, junto con la práctica 14, incluida en los apuntes: https://aulavirtual.educa.jcyl.es/iesriberaadecastilla/pluginfile.php/73491/mod_folder/content/0/2_DAM_SGE_UD_4_Entorno_desarrollo_y_primer_modulo_odoo.pdf?forcedownload=1
- Odoo. Descripción general de la arquitectura: https://www.odoo.com.translate.goog/documentation/18.0/developer/tutorials/server_framework_101/01_arhitecture.html?x_tr_sl=en&x_tr_tl=es&x_tr_hl=es&x_tr_pto=rq
- Viewnext. Cambios cronológicos en la guía de Scrum: <https://viewnext.usal.es/blog/cambios-cronológicos-en-la-guía-de-scrum>
- Asana. ¿Qué es Scrum?: <https://asana.com/es/resources/what-is-scrum>
- Principales conceptos de Scrum. Tema 8 del Moodle: https://aulavirtual.educa.jcyl.es/iesriberaadecastilla/pluginfile.php/74204/mod_folder/content/0/2_DAM_SGE_UD_8_modulo_manage.pdf?forcedownload=1