



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: 2024/2025

Fecha de presentación: 22/12/2024

Nombre y Apellidos: Rebeca Martínez de León
Email: rebeca.marleo@educa.jcyl.es

Índice

Modificar la **portada** con vuestros datos personales.

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

El primer apartado que vamos a incluir es la **INTRODUCCIÓN**: hablar brevemente sobre los sistemas ERP y las metodologías ágiles (en concreto sobre SCRUM).

Segundo apartado: ORGANIZACIÓN DE LA MEMORIA (lo vamos a dejar para el final; consistirá en enumerar los apartados de la memoria, con una pequeña descripción de cada uno de ellos).

Tercer apartado: ESTADO DEL ARTE. Con los siguientes subapartados:

1. ERP

- a. Definición de los ERP
- b. Evolución de los ERPs
- c. Principales ERP
- d. ERP seleccionado (Odoo)
- e. Instalación y desarrollo (*formas de instalación, explicando la que se va a usar para desarrollar el proyecto: Docker*)
- f. Especificaciones técnicas
 - i. Arquitectura de Odoo
 - ii. Composición de un módulo

2. SCRUM

- a. Definición de SCRUM
- b. Evolución
- c. Funcionamiento
- d. Principales conceptos (*explicar los principales conceptos: proyecto, historias de usuario, sprint, tarea...*)

Enlace sugerido: <https://www.atlassian.com/es/agile/scrum>

CONTINUACIÓN PROYECTO MANAGE

Investigar una posible ampliación de este proyecto e implementarla en vuestra aplicación. La ampliación tiene que incluir alguno(s) de los siguientes apartados:

- Algún aspecto no visto hasta ahora en el módulo de Sistemas de Gestión Empresarial
- Algún aspecto relacionado con el uso de CRUD, usando el ORM de Odoo

Añadir los siguientes puntos, a continuación de los que ya teníamos anteriormente en la memoria:

Descripción general del proyecto:

- **Objetivos** (breve descripción de lo que se ha pretendido alcanzar con el proyecto);
- **Entorno de trabajo** (explicar todas las herramientas utilizadas para desarrollar el proyecto: Docker, navegador, visual studio code...)

Diseño de la aplicación:

- **Modelo relacional de la BBDD**
- **Partes del proyecto** (models, views, security...), explicando brevemente lo que consideréis importante
- **Ampliación del proyecto**, explicando detalladamente el objetivo de la ampliación, el desarrollo...

Pruebas de funcionamiento

Conclusiones y posibles ampliaciones

Bibliografía

Normas de entrega:

Extensión mínima: 30 páginas

Formato entrega:

- Repositorio de Github, con los archivos generados en el desarrollo.
- Archivo README con este contenido: título del proyecto, descripción del proyecto, enlace a la memoria del proyecto en formato pdf
- La memoria en archivo independiente,
 - formato pdf;
 - nombre: Apellido1_Apellido2_Nombre_proyectomanage.pdf

Presentación oral en clase de algunos apartados del proyecto:

- Descripción general del proyecto
- Diseño de la aplicación (incluir vuestra ampliación)
- Conclusiones y posibles ampliaciones

Criterios calificación memoria proyecto manage:

Presentación formal (20%)

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

Contenidos (40%)

- Originalidad del tema elegido (contenidos originales, no repetidos)

- Grado de dificultad en orden a la investigación de contenidos
- Grado de profundización en la investigación de contenidos
- Explicación clara y concisa, con capturas, explicaciones breves...
- Incluye todos los apartados requeridos en el proyecto

Defensa oral (40%)

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

Introducción

Los sistemas ERP simplifican enormemente las tareas repetitivas que se deben realizar en una empresa. Su funcionamiento sencillo e intuitivo y su complejidad interna las convierte en las herramientas perfectas para el manejo de las labores del trabajo.

Por su lado, las metodologías ágiles consisten en la división de las tareas en distintos equipos enfocados en cada una de ellas de forma individual. El método Scrum sigue esta filosofía y está enfocado a la realización de tareas en cortos periodos de tiempo y con gran probabilidad de cambios a lo largo del proyecto. Es por esto que el método Scrum es conocido como “la metodología del caos”.

Organización de la memoria

La memoria está dividida en 3 apartados principales.

El primer apartado es la introducción. En él se presentan brevemente los sistemas ERP y las metodologías ágiles, en particular Scrum.

En el segundo apartado se resumen el resto de apartados.

Por último, en el tercer apartado se explica el estado del arte en cuestión de ERP y Scrum. Este apartado se divide a su vez en otros subapartados.

Primero se habla de los ERP, comenzando por una definición del término y continuando con su evolución. Después se habla de los principales ERP, del ERP seleccionado para realizar el proyecto y de la instalación y desarrollo en dicho ERP. Por último, se mencionan sus especificaciones técnicas, que son la arquitectura de Odoo y la composición de un módulo.

Luego se explica Scrum, también iniciando con una definición del mismo y su evolución a lo largo de los años. Además, se explica su funcionamiento y sus principales conceptos.

Estado del arte

1. ERP

a) Definición de los ERP

Un ERP o sistema de planificación de recursos empresariales es un conjunto de programas que integra todos los datos y procesos de una organización en un solo sistema unificado para la realización de las operaciones internas más importantes de la empresa de forma automática, permitiendo la trazabilidad de todos sus procesos y dando así paso a la planificación y optimización de los recursos.

b) Evolución de los ERPs

1960: En esta década se produjo el surgimiento de los MRP o sistemas de planificación de requerimientos de materiales, los cuales se utilizaban para la gestión de inventario y procesos de producción. Estos sistemas tenían grandes limitaciones, ya que estaban principalmente enfocados al sector manufacturero y su instalación requería una gran inversión en hardware. A pesar de ello, tuvieron un gran impacto debido a los beneficios que proporcionaban en términos de eficiencia y asentaron las bases de lo que estaría por venir.

1970: Durante los 70, los sistemas MRP que habían surgido en la década anterior se fueron perfeccionando. Así adquirieron nuevas funcionalidades como la planificación de la producción, la programación de la planta o la gestión de capacidades. Los MRP eran cada vez más populares y cada vez más empresas contaban con ellos.

1980: La década de los 80 fue el momento clave en que los MRP evolucionaron para dar lugar a los ERP. Estos nuevos sistemas contaban con características más allá de la planificación de la producción, como manejo de finanzas, recursos humanos, ventas o distribución. Los ERP supusieron una revolución para las empresas, ya que les permitieron tener una visión unificada de todos los procesos de su negocio.

1990: En los años 90 los ERP se expandieron a una gran variedad de industrias, incluyendo a medianas y pequeñas empresas, ya que durante esta década estos sistemas se volvieron más asequibles debido a la reducción del costo de hardware. Comenzó la creación de módulos cada vez más especializados para la labor de cada empresa. Durante este periodo se produjo también el surgimiento de internet y con él el inicio de la globalización, permitiendo que los ERP ayudasen a la coordinación de tareas a nivel global.

2000: Con la llegada del nuevo milenio se introdujeron los nuevos ERP en la nube, los cuales permitieron a las empresas acceder a sus sistemas de planificación desde internet, reduciendo enormemente los costos de infraestructuras y mantenimiento de las mismas y aumentando aún más su accesibilidad a las empresas de menor tamaño. Además, se implementó una nueva forma de adquirir estos servicios; en lugar de pagar el precio completo una sola vez se creó un modelo de suscripciones. En esta década se empezaron a integrar nuevas tecnologías como el big data, el internet de las cosas o la inteligencia artificial.

2010: Durante la década del 2010 las empresas solicitaron poder acceder a sus ERP desde cualquier parte por lo que se desarrollaron aplicaciones móviles. También hubo un enorme auge en la personalización de estos sistemas. Las empresas ya no se conformaban con tener soluciones genéricas y querían módulos adaptados a sus respectivas necesidades.

2020: En la actualidad, los ERP siguen creciendo con un fuerte enfoque en el uso de inteligencia artificial, el análisis de datos avanzado y la automatización de procesos. Estas tecnologías permiten a las empresas analizar información de forma masiva y predecir tendencias que les ayuden a la toma de decisiones. Es destacable además la mejora en las interfaces, siendo cada vez más intuitivas y personalizables para la mayor comodidad del usuario, incrementando así la productividad.

c) Principales ERP

En la actualidad contamos con un amplio abanico de sistemas ERP entre los cuales elegir, pero los principales son los siguientes:

- ERP Oracle Fusion Cloud
- ERP Microsoft Dynamics 365
- Workday Enterprise Management Cloud
- Oracle NetSuite
- SAP S/ 4HANA

d) ERP seleccionado

Entre los ERP disponibles, Odoo ha sido el ERP seleccionado para realizar este proyecto, ya que este se puede considerar una alternativa de código abierto muy sólida a sistemas ERP de alta categoría como Microsoft Dynamics o SAP.

e) Instalación y desarrollo

Para instalar Odoo debemos crearnos una cuenta en su página, después seleccionamos la versión de Odoo que deseemos y seguimos los pasos que nos indique el installer.

Para desarrollar un módulo en Odoo se utiliza el lenguaje Python. Gracias al comando scaffold podemos crear una estructura básica para nuestro proyecto que iremos modificando hasta crear el módulo personalizado de Odoo que queramos. Este comando se ejecuta en la terminal de Docker, el programa que emplearemos para manejar las bases de datos de nuestros módulos.

No existe un programa específico para programar módulos de Odoo por lo que podemos usar nuestro editor de texto de preferencia, en este caso Visual Studio Code.

f) Especificaciones técnicas

i. Arquitectura de Odoo

La arquitectura de Odoo está separada en varios niveles, siendo estos la presentación, la lógica empresarial y el almacenamiento de datos.

El nivel de la presentación es una combinación de HTML5, JavaScript y CSS.

El nivel lógico está escrito en Python puro.

El nivel de datos solo admite PostgreSQL.

ii. Composición de un módulo

Un módulo de Odoo está compuesto por los siguientes elementos, aunque ninguno de ellos es obligatorio:

- Objetos de negocio: son clases Python en las que se declara la información de los modelos del módulo.
- Vistas de objetos: define la interfaz de usuario.
- Archivos de información: los archivos XML o CSV que declaran los datos del modelo.
- Controladores web: manejan las solicitudes de los navegadores web.
- Datos web estáticos: las imágenes, archivos CSS o JavaScript utilizados por la interfaz web o el sitio web.

2. Scrum

a) Definición de Scrum

Scrum es un marco de gestión de proyectos ágil que ayuda a equipos a estructurar y gestionar el trabajo conjunto gracias a los valores, principios y prácticas característicos de la metodología, que anima a estos equipos a aprender de sus experiencias, a organizarse mientras trabajan en solventar un problema y a reflexionar tanto sobre sus victorias como sus derrotas y aprender de ambas.

b) Evolución

1986: es el año en el que nace esta metodología. Sus creadores, Ikujiro Nonaka e Hirotaka Takeuchi, se basaron en un estudio realizado en distintas empresas que estaban buscando un nuevo enfoque de trabajo. Estas empresas estaban basando su producción en equipos de trabajo autoorganizados y enfatizaba la importancia de esta autonomía.

1993: Jeff Sutherland y su equipo crean el proceso de Scrum para desarrollo de software combinando los conceptos del artículo de 1986 con los de desarrollo orientado a objetos, control de procesos empírico, desarrollo iterativo incremental, proceso de software y mejora de la productividad, así como el desarrollo de sistemas complejos y dinámicos.

1995: Jeff Sutherland y Ken Schwaber crean un conjunto de reglas y buenas prácticas enfocadas al desarrollo de software y lo bautizan con el nombre de Scrum.

2001: tras la publicación del Manifiesto Ágil la popularidad de Scrum aumentó.

2020: la guía Scrum recibe una actualización importante para hacer la práctica más accesible, sencilla y abierta para entornos fuera del campo de las tecnologías de la información.

c) Funcionamiento

La metodología Scrum consiste en abordar un proyecto dividiéndolo en sprints o partes más pequeñas. Dentro de este entorno de trabajo hay que seguir una serie de fases para abordar cada

tarea, y participan unos roles específicos que garantizan el cumplimiento de esta filosofía de trabajo.

d) Principales conceptos

Los principales conceptos sobre los que se fundamenta la metodología Scrum son los siguientes:

- Proyecto: es el trabajo general que el equipo va a desarrollar. Suelen planificarse completamente desde el principio y su estructura se define a través de las funcionalidades y las metas a cumplir.
- Sprints: son ciclos cortos y repetitivos que suelen durar entre 1 o 4 semanas. En cada sprint, el equipo se enfoca en completar un conjunto de tareas definidas para entregar una versión funcional del producto.
- Historias de usuario: son la manera en que Scrum representa los requisitos o funcionalidades del proyecto desde la perspectiva del usuario final. Normalmente se escriben en un formato específico y deben ser breves, así ayudan a enfocarse en las necesidades y beneficios reales del usuario.
- Tarea: es una división más pequeña de trabajo que deriva de una historia de usuario y representa un paso concreto necesario para completar esta. Deben ser muy específicas y breves, pues su duración óptima es de a penas unas horas o un día como máximo.
- Equipos auto-organizados y multifuncionales: son pequeños, de entre 5 a 9 personas y cada miembro aporta habilidades específicas, trabajando de manera colaborativa sin una jerarquía fija.

Descripción general del proyecto

Objetivos

El objetivo de este proyecto es crear un módulo de odoo que sirva para la organización del trabajo mediante la metodología ágil Scrum, permitiendo la creación y manejo de proyectos, sprints, historias, tecnologías, desarrolladores y tareas.

Entorno de trabajo

Para la realización del proyecto se utilizaron diversas herramientas con el fin de llevar a cabo la creación, despliegue y pruebas de funcionamiento del módulo a desarrollar. A continuación, se explican en profundidad estos recursos:

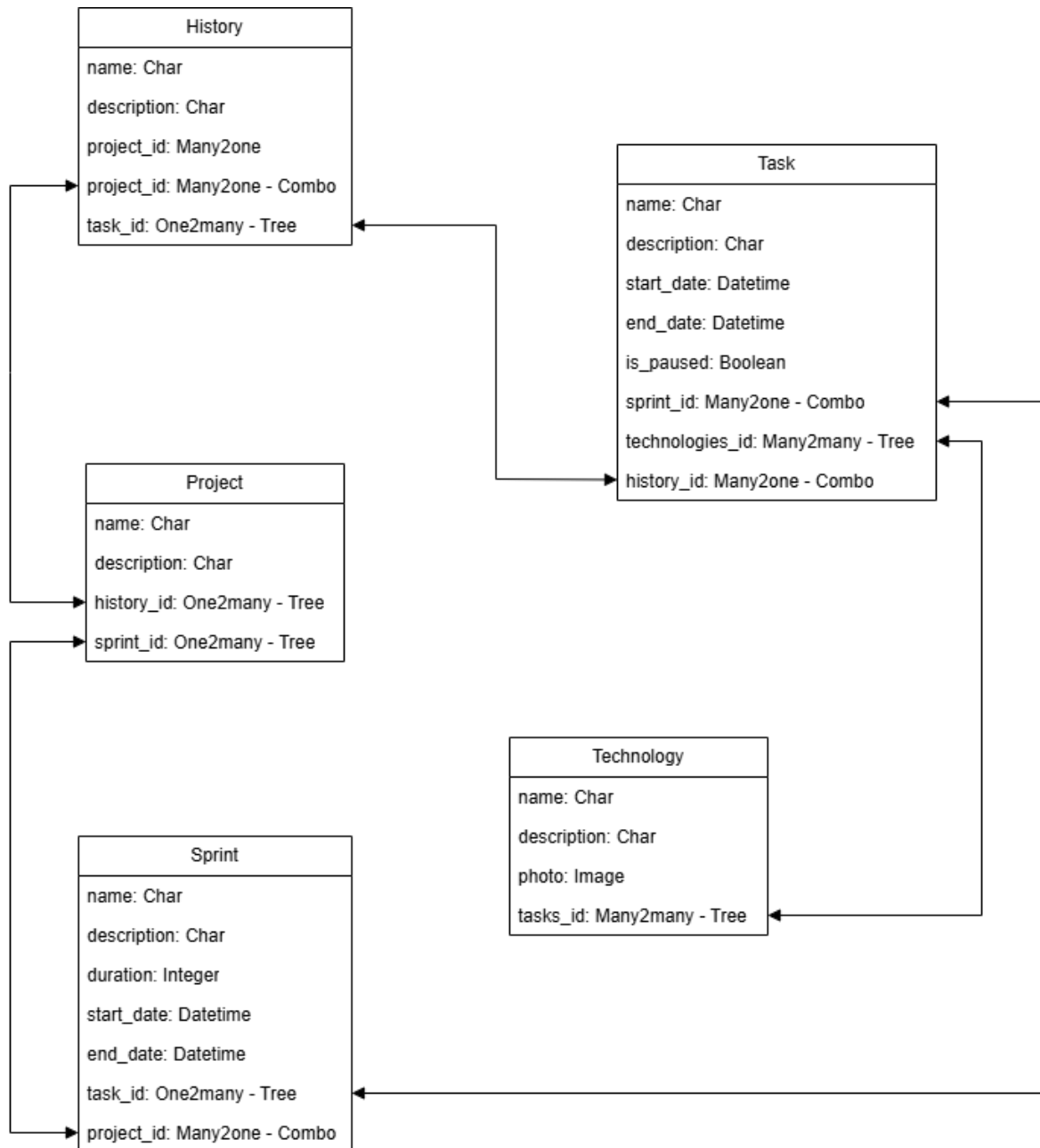
- Visual Studio Code:

Este fue el editor de texto escogido para la creación y modificación de los archivos requeridos para el correcto funcionamiento del módulo. Los principales tipos de archivos utilizados fueron:

- .py: esta extensión corresponde a los modelos que definen la funcionalidad del módulo.
 - .xml: son los archivos que definen las vistas del módulo.
 - .csv: corresponde al archivo de seguridad que da los permisos necesarios al resto de archivos para poder realizar sus respectivas funciones.
-
- Google Chrome: como navegador en el que desplegar, en un entorno localhost, y comprobar los cambios realizados en el módulo. Gracias a esta herramienta se pudieron comprobar posibles fallos que causen errores de servidor o de incompatibilidad entre las diferentes partes del programa, el aspecto de este y su correcto funcionamiento de acuerdo a lo que se espera de un sistema Scrum.
 - Docker: la plataforma para automatizar el despliegue del módulo dentro de los contenedores, garantizando que cada servicio tenga sus propias dependencias y configuraciones y manejo de la base de datos del módulo.

Diseño de la aplicación

Modelo relacional de la BBDD



Partes del proyecto

Modelos

- Modelo developer.py:

```
developer.py 1 X
models > developer.py > ...
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4
5
6  class developer(models.Model):
7      _name = 'res.partner'
8      _inherit = 'res.partner'
9
10     is_dev = fields.Boolean(default = True)
11
12     technologies = fields.Many2many("managerebeca.technology",
13                                     relation="developer_technologies",
14                                     column1="developer_id",
15                                     column2="technologies_id")
16
17     @api.model
18     def create(self, vals):
19         # asegura que el campo is_dev sea True al crear un nuevo desarrollador
20         vals['is_dev'] = True
21         return super(developer, self).create(vals)
```

El modelo developer.py hereda del módulo res.partner mediante su atributo `_inherit`.

```
8      _inherit = 'res.partner'
```

Tiene un atributo que se establece como True por defecto para ver si los nuevos developers añadidos desde su formulario correspondiente son desarrolladores.

```
10     is_dev = fields.Boolean(default = True)
```

Además, tiene una relación en la que un desarrollador puede usar una o más tecnologías diferentes y a su vez esas tecnologías pueden ser utilizadas por más de un desarrollador.

```
12     technologies = fields.Many2many("managerebeca.technology",
13                                     relation="developer_technologies",
14                                     column1="developer_id",
15                                     column2="technologies_id")
```

La función create asegura que al crearse un nuevo desarrollador el valor del campo is_dev se establezca como verdadero.

```
17     @api.model
18     def create(self, vals):
19         # asegura que el campo is_dev sea True al crear un nuevo desarrollador
20         vals['is_dev'] = True
21         return super(developer, self).create(vals)
```

- Modelo history.py:

```
history.py 1 x
models > history.py > ...
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4
5
6  class history(models.Model):
7      _name = 'managerebeca.history'
8      _description = 'managerebeca.history'
9
10     name = fields.Char(String="Nombre")
11     description = fields.Char(String="Descripción")
12
13     project_id = fields.Many2one(comodel_name="managerebeca.project", string = "Proyecto", ondelete="set null", inverse_name = "histories_ids")
14     tasks_ids = fields.One2many(string = "Tareas", comodel_name = "managerebeca.task", inverse_name = "history_id")
15     used_technologies = fields.Many2many("managerebeca.technology", compute = "_get_used_technologies")
16
17     def _get_used_technologies(self):
18         for history in self:
19             technologies = None # Array para concatenar todas las tecnologías. Inicialmente no tiene valor
20             for task in history.tasks_ids: # Para cada una de las tareas de la historia
21                 if not technologies:
22                     technologies = task.technologies_ids
23                 else:
24                     technologies = technologies + task.technologies_ids
25             history.used_technologies = technologies # Asignar las tecnologías a la historia
```

El modelo history.py es un modelo nativo del módulo managerebeca.

Tiene una relación en la que una historia solo puede pertenecer a un proyecto pero un proyecto puede tener múltiples historias.

```
13     project_id = fields.Many2one("managerebeca.project", string = "Proyecto", ondelete="set null")
```

Además, una historia puede tener múltiples tareas pero una tarea solo puede pertenecer a una historia.

```
14     tasks_ids = fields.One2many(string = "Tareas", comodel_name = "managerebeca.task", inverse_name = "history_id")
```

En una historia se pueden utilizar múltiples tecnologías diferentes y a su vez cada una de esas tecnologías pueden ser utilizadas en muchas tareas diferentes. Este campo se autocompleta mediante el método `_get_used_technologies`, mediante el cual se recogen todas las tecnologías usadas en las tareas en una lista.

```
15     used_technologies = fields.Many2many("managerebeca.technology", compute = "_get_used_technologies")
```

```

17     def _get_used_technologies(self):
18         for history in self:
19             technologies = None # Array para concatenar todas las tecnologías. Inicialmente no tiene valor
20             for task in history.tasks_ids: # Para cada una de las tareas de la historia
21                 if not technologies:
22                     technologies = task.technologies_ids
23                 else:
24                     technologies = technologies + task.technologies_ids
25             history.used_technologies = technologies # Asignar las tecnologías a la historia

```

- Modelo project.py:

```

project.py 1 x
models > project.py > ...
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4
5
6  class project(models.Model):
7      _name = 'managerebeca.project'
8      _description = 'managerebeca.project'
9
10     name = fields.Char(String="Nombre")
11     description = fields.Char(String="Descripción")
12
13     histories_ids = fields.One2many("managerebeca.history", string = "Historia", inverse_name = "project_id")
14     sprints_ids = fields.One2many(comodel_name = "managerebeca.sprint", inverse_name = "project_id", string = "Sprint")

```

El modelo project.py es un modelo nativo del módulo managerebeca.

Tiene una relación en la que un proyecto puede tener múltiples historias pero una historia solo puede pertenecer a un proyecto.

```

13     histories_ids = fields.One2many("managerebeca.history", string = "Historia", inverse_name = "project_id")

```

Además, un proyecto puede tener múltiples sprints pero un sprint solo puede pertenecer a un proyecto.

```

14     sprints_ids = fields.One2many(comodel_name = "managerebeca.sprint", inverse_name = "project_id", string = "Sprint")

```

- Modelo sprint.py:

```
sprint.py 1 X
models > sprint.py > ...
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4  import datetime
5
6
7  class sprint(models.Model):
8      _name = 'managerebeca.sprint'
9      _description = 'managerebeca.sprint'
10
11     name = fields.Char(string="Nombre")
12     description = fields.Text(string="Descripción")
13     duration = fields.Integer(default = 15)
14     start_date = fields.Datetime(string="Fecha inicio")
15     end_date = fields.Datetime(string="Fecha fin", compute="_get_end_date", store=True)
16
17     tasks_ids = fields.One2many(string = "Tareas", comodel_name = "managerebeca.task", inverse_name = "sprint_id")
18     project_id = fields.Many2one(comodel_name="managerebeca.project", inverse_name="sprints_ids", string="Proyectos")
19
20     @api.depends('start_date', 'duration')
21     def _get_end_date(self):
22         for sprint in self:
23             #try:
24             if isinstance(sprint.start_date, datetime.datetime) and sprint.duration > 0:
25                 sprint.end_date = sprint.start_date + datetime.timedelta(days=sprint.duration)
26             else:
27                 sprint.end_date = sprint.start_date
```

El modelo sprint.py es un modelo nativo del módulo managerebeca.

Tiene un atributo que se establece como 15 por defecto para establecer la duración base de un sprint que luego podrá ser modificada.

```
12     description = fields.Text(string="Descripción")
13     duration = fields.Integer(default = 15)
14     start_date = fields.Datetime(string="Fecha inicio")
```

Su atributo start_date determina la fecha de inicio del sprint y su atributo end_date determina la fecha de finalización. Este último se calcula a partir de la función _get_end_date, que toma la fecha de inicio y la duración y las suma para obtener la fecha de finalización.

```
14     start_date = fields.Datetime(string="Fecha inicio")
15     end_date = fields.Datetime(string="Fecha fin", compute="_get_end_date", store=True)
16
17     @api.depends('start_date', 'duration')
21     def _get_end_date(self):
22         for sprint in self:
23             #try:
24             if isinstance(sprint.start_date, datetime.datetime) and sprint.duration > 0:
25                 sprint.end_date = sprint.start_date + datetime.timedelta(days=sprint.duration)
26             else:
27                 sprint.end_date = sprint.start_date
```

Tiene una relación en la que un sprint puede tener múltiples tareas pero una tarea solo puede pertenecer a un sprint.


```
17 tasks_ids = fields.One2many(string = "Tareas", comodel_name = "managerebeca.task", inverse_name = "sprint_id")
```

Además, tiene una relación en la que un sprint solo puede pertenecer a un proyecto pero un proyecto puede tener múltiples sprints.

```
18 project_id = fields.Many2one(comodel_name="managerebeca.project", inverse_name="sprints_ids", string="Proyectos")
```

- Modelo task.py:

```
task.py 1 x
models > task.py > ...
1 # -*- coding: utf-8 -*-
2
3 import datetime
4 from odoo import models, fields, api
5
6
7 class task(models.Model):
8     _name = 'managerebeca.task'
9     _description = 'managerebeca.task'
10
11     name = fields.Char(String="Nombre")
12     description = fields.Char(String="Descripción")
13     start_date = fields.Datetime()
14     end_date = fields.Datetime()
15     is_paused = fields.Boolean()
16
17     sprint_id = fields.Many2one(comodel_name="managerebeca.sprint", compute = "_get_sprint", store = True, inverse_name="tasks_ids")
18     technologies_ids = fields.Many2many(comodel_name = "managerebeca.technology",
19                                       relation = "technologies_tasks",
20                                       column1 = "technologies_ids",
21                                       column2 = "tasks_ids",
22                                       string = "Tecnologías",)
23     history_id = fields.Many2one(comodel_name="managerebeca.history", ondelete="set null", help="Historia relacionada", inverse_name = "tasks_ids")
24     definition_date = fields.Datetime(default=lambda p: datetime.datetime.now())
25     project_id = fields.Many2one(comodel_name="managerebeca.project", related = "history_id.project_id", readonly = True)
26     code = fields.Char(String = "Código", compute = "_get_code")
27
28     @api.depends('start_date', 'end_date')
29     def _get_sprint(self):
30         for task in self:
31             sprints = self.env['managerebeca.sprint'].search([(('project_id', '=', task.history_id.project_id)])]
32             found = False
33             for sprint in sprints:
34                 if isinstance(sprint.end_date, datetime.datetime) and sprint.end_date > datetime.datetime.now():
35                     task.sprint_id = sprint.id
36                     found = True
37             if not found:
38                 task.sprint_id = False
39
40     @api.one
41     def _get_code(self):
42         for task in self:
43             # try:
44             task.code = "TSK_"+str(task.id)
45             #_logger.info("Código generado: "+task.code)
46             #except:
47             #raise ValidationError(_("Generación de código errónea"))
```

El modelo task.py es un modelo nativo del módulo managerebeca.

Tiene una relación en la que una tarea solo puede pertenecer a un sprint pero un sprint puede tener múltiples tareas. En este campo se utiliza la función `_get_sprint` para obtener los sprints que tengan una fecha de finalización posterior a la fecha actual. De esta manera se valida que las tareas no se puedan asociar a sprints que ya hayan terminado.

```
17 sprint_id = fields.Many2one(comodel_name="managerebeca.sprint", compute = "_get_sprint", store = True, inverse_name="tasks_ids")
```

```

28     @api.depends('start_date', 'end_date')
29     def _get_sprint(self):
30         for task in self:
31             sprints = self.env['managerebeca.sprint'].search([('project_id', '=', task.history_id.project_id.id)])
32             found = False
33             for sprint in sprints:
34                 if isinstance(sprint.end_date, datetime.datetime) and sprint.end_date > datetime.datetime.now():
35                     task.sprint_id = sprint.id
36                     found = True
37             if not found:
38                 task.sprint_id = False
39 
```

Además, una tarea puede usar una o más tecnologías diferentes y a su vez esas tecnologías pueden ser utilizadas por más de una tarea.

```

18     technologies_ids = fields.Many2many(comodel_name="managerebeca.technology",
19                                         relation="technologies_tasks",
20                                         column1="technologies_ids",
21                                         column2="tasks_ids",
22                                         string="Tecnologías",)
23 
```

También tiene una relación en la que una tarea solo puede pertenecer a una historia pero una historia puede tener múltiples tareas.

```

23     history_id = fields.Many2one(comodel_name="managerebeca.history", ondelete="set null", help="Historia relacionada", inverse_name="tasks_ids")
24 
```

Tiene un atributo que se establece por defecto como la fecha y hora actuales que define la hora de creación de la tarea.

```

24     definition_date = fields.Datetime(default=lambda p: datetime.datetime.now())
25 
```

Además, tiene una relación en la que una tarea solo puede pertenecer a un proyecto pero un proyecto puede tener múltiples tareas.

```

25     project_id = fields.Many2one(comodel_name="managerebeca.project", related="history_id.project_id", readonly=True)
26 
```

También tiene un atributo que establece un código a la tarea. Este atributo se calcula automáticamente mediante el método `_get_code` que establece el código como una concatenación de la cadena TSK_ y el id correspondiente de dicha tarea.

```

26     code = fields.Char(String="Código", compute="_get_code")
27 
```

```

40     #@api.one
41     def _get_code(self):
42         for task in self:
43             # try:
44                 task.code = "TSK_"+str(task.id)
45                 #_logger.info("Código generado: "+task.code)
46             #except:
47                 #raise ValidationError(_("Generación de código errónea"))
48 
```

- Modelo technology.py:

```
technology.py 1 X
models > technology.py > ...
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4
5
6  class technology(models.Model):
7      _name = 'managerebeca.technology'
8      _description = 'managerebeca.technology'
9
10     name = fields.Char(String="Nombre")
11     description = fields.Char(String="Descripción")
12     photo = fields.Image(string="Imagen", max_width=200, max_height=200)
13
14     tasks_ids = fields.Many2many(comodel_name = "managerebeca.task",
15                                 relation = "technologies_tasks",
16                                 column1 = "tasks_ids",
17                                 column2 = "technologies_ids")
```

El modelo technology.py es un modelo nativo del módulo managerebeca.

Tiene un atributo photo para introducir una foto de la tecnología empleada.

```
12     photo = fields.Image(string="Imagen", max_width=200, max_height=200)
13
```

Tiene una relación en la que una tecnología puede usarse en una o más tareas diferentes y a su vez esas tareas pueden ser utilizadas por más de una tecnología.

```
14     tasks_ids = fields.Many2many(comodel_name = "managerebeca.task",
15                                 relation = "technologies_tasks",
16                                 column1 = "tasks_ids",
17                                 column2 = "technologies_ids")
```

Vistas

- Vista developer.xml:

```

1 <odoo>
2   <data>
3     <record model="ir.ui.view" id="managerebeca.devs_partner_form">
4       <field name="name">manage devs form</field>
5       <field name="model">res.partner</field>
6       <field name="inherit_id" ref="base.view_partner_form"></field>
7       <field name="mode">primary</field>
8       <field name="arch" type="xml">
9         <xpath expr="//sheet/notebook/page[@name='internal_notes']" position="after">
10           <page name="devs" string="Devs">
11             <group>
12               <group>
13                 <field name="technologies"></field>
14               </group>
15             </group>
16           </page>
17         </xpath>
18       </field>
19     </record>
20
21     <record model="ir.actions.act_window" id="managerebeca.action_developer_window">
22       <field name="name">manage developer window</field>
23       <field name="res_model">res.partner</field>
24       <field name="view_mode">tree,form</field>
25       <field name="domain">[('is_dev', '=', True)]</field>
26     </record>
27
28     <record model="ir.actions.act_window.view" id="managerebeca.action_view_developer_tree">
29       <field name="sequence" eval="1"></field>
30       <field name="view_mode">tree</field>
31       <field name="view_id" ref="base.view_partner_tree"></field>
32       <field name="act_window_id" ref="managerebeca.action_developer_window"></field>
33     </record>
34
35     <record model="ir.actions.act_window.view" id="managerebeca.action_view_developer_form">
36       <field name="sequence" eval="2"></field>
37       <field name="view_mode">form</field>
38       <field name="view_id" ref="managerebeca.devs_partner_form"></field>
39       <field name="act_window_id" ref="managerebeca.action_developer_window"></field>
40     </record>
41
42     <menuitem name="Devs" id="menu_managerebeca_developer" parent="menu_managerebeca_management"
43       action="managerebeca.action_developer_window" />
44   </data>
45 </odoo>

```

La vista developer.xml hereda de la vista del formulario y la vista tree del módulo res.partner mediante el atributo ref="base.view_partner_form" y ref="base.view_partner_tree" respectivamente.

```

5       <field name="model">res.partner</field>
6       <field name="inherit_id" ref="base.view_partner_form"></field>
7       <field name="mode">primary</field>
...
31      <field name="view_id" ref="base.view_partner_tree"></field>

```

Además, añade una nueva página a las pestañas específicas por asignación de la parte de abajo del formulario, la página devs con su respectivo campo technologies.

```
9      <xpath expr="//sheet/notebook/page[@name='internal_notes']" position="after">
10      <page name="devs" string="Devs">
11      <group>
12      <group>
13      <field name="technologies"></field>
14      </group>
15      </group>
16      </page>
17      </xpath>
```

La vista developer.xml se puede acceder mediante su id menu_managerebeca_developer que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```
42      <menuitem name="Devs" id="menu_managerebeca_developer" parent="menu_managerebeca_management"
```

- Vista history.xml:

```

history.xml X
views > history.xml > odoo > data > record > field > tree > field
1  <odoo>
2    <data>
3      <!-- explicit list view definition -->
4      <record model="ir.ui.view" id="vista_managerebeca_history_tree">
5        <field name="name">vista_managerebeca_history_tree</field>
6        <field name="model">managerebeca.history</field>
7        <field name="arch" type="xml">
8          <tree>
9            <field name="name" />
10           <field name="description" />
11         </tree>
12       </field>
13     </record>
14
15     <record model="ir.ui.view" id="vista_managerebeca_history_form">
16       <field name="name">vista_managerebeca_history_form</field>
17       <field name="model">managerebeca.history</field>
18       <field name="arch" type="xml">
19         <form string="formulario_history">
20           <sheet>
21             <group name="group_top">
22               <field name="name" />
23               <field name="description" />
24               <field name="project_id" />
25               <field name="tasks_ids" />
26               <field name="used_technologies" />
27             </group>
28           </sheet>
29         </form>
30       </field>
31     </record>
32
33     <record model="ir.actions.act_window" id="accion_managerebeca_history_form">
34       <field name="name">Listado de historias</field>
35       <field name="type">ir.actions.act_window</field>
36       <field name="res_model">managerebeca.history</field>
37       <field name="view_mode">tree,form</field>
38       <field name="help" type="html">
39         <p class="oe_view_nocontent_create">
40           Historial
41         </p>
42         <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
43       </p>
44     </field>
45   </record>
46
47   <!-- actions -->
48
49   <menuitem name="Histories" id="menu_managerebeca_history" parent="menu_managerebeca_management"
50     action="accion_managerebeca_history_form" />
51
52 </data>
53 </odoo>

```

La vista tree toma el modelo history de managerebeca y muestra los campos del nombre y la descripción de la historia.

```

6      <field name="model">managerebeca.history</field>

```

```

9         <field name="name" />
10        <field name="description" />
11    </tree>

```

La vista form toma el modelo history de managerebeca y recibe información para los campos nombre, descripción, proyecto, tareas y tecnologías usadas.

```

17        <field name="model">managerebeca.history</field>
18        <field name="arch" type="xml">
19
20            <field name="name" />
21            <field name="description" />
22            <field name="project_id" />
23            <field name="tasks_ids" />
24            <field name="used_technologies" />
25        </field>

```

La vista history.xml se puede acceder mediante su id menu_managerebeca_history que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```

49        <menuitem name="Histories" id="menu_managerebeca_history" parent="menu_managerebeca_management"
50        </menuitem>

```

- Vista project.xml:

```

1 <!-- explicit list view definition -->
2 <record model="ir.ui.view" id="vista_managerebeca_project_tree">
3   <field name="name">vista_managerebeca_project_tree</field>
4   <field name="model">managerebeca.project</field>
5   <field name="arch" type="xml">
6     <tree>
7       <field name="name" />
8       <field name="description" />
9     </tree>
10  </field>
11 </record>
12
13 <record model="ir.ui.view" id="vista_managerebeca_project_form">
14   <field name="name">vista_managerebeca_project_form</field>
15   <field name="model">managerebeca.project</field>
16   <field name="arch" type="xml">
17     <form string="formulario_project">
18       <sheet>
19         <group name="group_top">
20           <field name="name" />
21           <field name="description" />
22           <field name="histories_ids" />
23           <field name="sprints_ids" />
24         </group>
25       </sheet>
26     </form>
27   </field>
28 </record>
29
30 <record model="ir.actions.act_window" id="accion_managerebeca_project_form">
31   <field name="name">Listado de proyectos</field>
32   <field name="type">ir.actions.act_window</field>
33   <field name="res_model">managerebeca.project</field>
34   <field name="view_mode">tree,form</field>
35   <field name="help" type="html">
36     <p class="oe_view_nocontent_create">
37       Listado de proyectos
38     </p>
39     <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
40   </field>
41 </record>
42
43 <!-- actions -->
44
45 <menuitem name="Projects" id="menu_managerebeca_project" parent="menu_managerebeca_management"
46   action="accion_managerebeca_project_form" />
47
48 </data>
49 </odoo>

```

La vista tree toma el modelo project de managerebeca y muestra los campos del nombre y la descripción del proyecto.

```

9   <field name="name" />
10  <field name="description" />

```


La vista form toma el modelo project de managerebeca y recibe información para los campos nombre, descripción, historias y sprints.

```
22 <field name="name" />
23 <field name="description" />
24 <field name="histories_ids" />
25 <field name="sprints_ids" />
26 </group>
```

La vista project.xml se puede acceder mediante su id menu_managerebeca_project que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```
48 <menuitem name="Projects" id="menu_managerebeca_project" parent="menu_managerebeca_management"
49 <action name="managerebeca_project_form" />
```

- Vista sprint.xml:

```
sprint.xml X
views > sprint.xml > odoo
1 <!-- explicit list view definition -->
2 <record model="ir.ui.view" id="vista_managerebeca_sprint_tree">
3   <field name="name">vista_managerebeca_sprint_tree</field>
4   <field name="model">managerebeca.sprint</field>
5   <field name="arch" type="xml">
6     <tree>
7       <field name="name" />
8       <field name="description" />
9     </tree>
10   </field>
11 </record>
12
13 <record model="ir.ui.view" id="vista_managerebeca_sprint_form">
14   <field name="name">vista_managerebeca_sprint_form</field>
15   <field name="model">managerebeca.sprint</field>
16   <field name="arch" type="xml">
17     <form string="formulario_sprint">
18       <sheet>
19         <group name="group_top">
20           <field name="name" />
21           <field name="description" />
22           <field name="duration" />
23           <field name="start_date" />
24           <field name="end_date" />
25           <field name="tasks_ids" />
26           <field name="project_id" />
27         </group>
28       </sheet>
29     </form>
30   </field>
31 </record>
32
33 <record model="ir.actions.act_window" id="accion_managerebeca_sprint_form">
34   <field name="name">Listado de sprints</field>
35   <field name="type">ir.actions.act_window</field>
36   <field name="res_model">managerebeca.sprint</field>
37   <field name="view_mode">tree,form</field>
38   <field name="help" type="html">
39     <p class="oe_view_nocontent_create">
40       Listado de sprints
41     </p>
42     <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
43     </p>
44   </field>
45 </record>
46
47 <!-- actions -->
48
49 <menuitem name="Sprints" id="menu_managerebeca_sprint" parent="menu_managerebeca_management"
50   action="accion_managerebeca_sprint_form" />
51
52 </data>
53
54 </odoo>
```

La vista tree toma el modelo sprint de managerebeca y muestra los campos del nombre y la descripción del sprint.

```
8      </tree>
9      <field name="name" />
10     <field name="description" />
11     </tree>
```

La vista form toma el modelo sprint de managerebeca y recibe información para los campos nombre, descripción, duración, fecha de inicio, fecha de finalización, tareas y proyecto.

```
22     <field name="name" />
23     <field name="description" />
24     <field name="duration" />
25     <field name="start_date" />
26     <field name="end_date" />
27     <field name="tasks_ids" />
28     <field name="project_id" />
29     </form>
```

La vista sprint.xml se puede acceder mediante su id menu_managerebeca_sprint que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```
51     <menuitem name="Sprints" id="menu_managerebeca_sprint" parent="menu_managerebeca_management"
52     </menuitem>
```

- Vista task.xml:

```
task.xml X
views > task.xml > odoo
1 <odoo>
2   <data>
3     <!-- explicit list view definition -->
4     <record model="ir.ui.view" id="vista_managerebeca_task_tree">
5       <field name="name">vista_managerebeca_task_tree</field>
6       <field name="model">managerebeca.task</field>
7       <field name="arch" type="xml">
8         <tree>
9           <field name="name" />
10          <field name="description" />
11        </tree>
12      </field>
13    </record>
14
15    <record model="ir.ui.view" id="vista_managerebeca_task_form">
16      <field name="name">vista_managerebeca_task_form</field>
17      <field name="model">managerebeca.task</field>
18      <field name="arch" type="xml">
19        <form string="formulario_task">
20          <sheet>
21            <group name="group_top">
22              <field name="code" />
23              <field name="name" />
24              <field name="description" />
25              <field name="start_date" />
26              <field name="end_date" />
27              <field name="is_paused" />
28              <field name="sprint_id" />
29              <field name="history_id" />
30              <field name="technologies_ids" />
31              <field name="definition_date" />
32            </group>
33          </sheet>
34        </form>
35      </field>
36    </record>
37
38    <record model="ir.actions.act_window" id="accion_managerebeca_task_form">
39      <field name="name">Listado de tareas</field>
40      <field name="type">ir.actions.act_window</field>
41      <field name="res_model">managerebeca.task</field>
42      <field name="view_mode">tree,form</field>
43      <field name="help" type="html">
44        <p class="oe_view_nocontent_create">
45          Listado de tareas
46        </p>
47        <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
48        </p>
49      </field>
50    </record>
51
52    <!-- actions -->
53
54    <menuitem name="Tasks" id="menu_managerebeca_task" parent="menu_managerebeca_management"
55      action="accion_managerebeca_task_form" />
56
57  </data>
58 </odoo>
```

La vista tree toma el modelo task de managerebeca y muestra los campos del nombre y la descripción de la tarea.

```

9      <field name="name" />
10     <field name="description" />

```

La vista form toma el modelo task de managerebeca y recibe información para los campos código, nombre, descripción, fecha de inicio, fecha de finalización, en pausa, sprint, historia, tecnologías y fecha de creación.

```

22     <field name="code" />
23     <field name="name" />
24     <field name="description" />
25     <field name="start_date" />
26     <field name="end_date" />
27     <field name="is_paused" />
28     <field name="sprint_id" />
29     <field name="history_id" />
30     <field name="technologies_ids" />
31     <field name="definition_date" />

```

La vista task.xml se puede acceder mediante su id menu_managerebeca_task que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```

54     <menuitem name="Tasks" id="menu_managerebeca_task" parent="menu_managerebeca_management"

```

- Vista technology.xml:

```

1 <!-- explicit list view definition -->
2 <record model="ir.ui.view" id="vista_managerebeca_technology_tree">
3   <field name="name">vista_managerebeca_technology_tree</field>
4   <field name="model">managerebeca.technology</field>
5   <field name="arch" type="xml">
6     <tree>
7       <field name="name" />
8       <field name="description" />
9     </tree>
10   </field>
11 </record>
12
13 <record model="ir.ui.view" id="vista_managerebeca_technology_form">
14   <field name="name">vista_managerebeca_technology_form</field>
15   <field name="model">managerebeca.technology</field>
16   <field name="arch" type="xml">
17     <form string="formulario_technology">
18       <sheet>
19         <group name="group_top">
20           <field name="name" />
21           <field name="description" />
22           <field name="photo" />
23           <field name="tasks_ids" />
24         </group>
25       </sheet>
26     </form>
27   </field>
28 </record>
29
30 <record model="ir.actions.act_window" id="accion_managerebeca_technology_form">
31   <field name="name">Listado de tecnologías</field>
32   <field name="type">ir.actions.act_window</field>
33   <field name="res_model">managerebeca.technology</field>
34   <field name="view_mode">tree,form</field>
35   <field name="help" type="html">
36     <p class="oe_view_nocontent_create">
37       Listado de tecnologías
38     </p>
39     <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
40     </p>
41   </field>
42 </record>
43
44 <!-- Top menu item -->
45 <menuitem name="Manage de Rebeca" id="menu_managerebeca_raiz" />
46
47 <!-- menu categories -->
48 <menuitem name="Management" id="menu_managerebeca_management" parent="menu_managerebeca_raiz" />
49
50 <!-- actions -->
51 <menuitem name="Technologies" id="menu_managerebeca_technology" parent="menu_managerebeca_management"
52   action="accion_managerebeca_technology_form" />
53
54 </data>
55 </odoo>

```

La vista tree toma el modelo technology de managerebeca y muestra los campos del nombre y la descripción de la tecnología.

```

9      <field name="name" />
10     <field name="description" />

```

La vista form toma el modelo tecnología de managerebeca y recibe información para los campos nombre, descripción, foto y tareas.

```

22     <field name="name" />
23     <field name="description" />
24     <field name="photo" />
25     <field name="tasks_ids" />

```

La vista technology.xml se puede acceder mediante su id menu_managerebeca_technology que, al igual que todas las vistas del módulo, cuelga del padre menu_managerebeca_management.

```

56     <menuitem name="Technologies" id="menu_managerebeca_technology" parent="menu_managerebeca_management"

```

En esta vista se establece el padre menu_managerebeca_management del que cuelgan todas las vistas. Este es a su vez hijo de menu_management_raiz.

```

52     <menuitem name="Management" id="menu_managerebeca_management" parent="menu_managerebeca_raiz" />

```

El padre de menu_managerebeca_management.

```

48     <menuitem name="Manage de Rebeca" id="menu_managerebeca_raiz" />

```

Archivos adicionales

- Archivo `__manifest__.py`:

```
__manifest__.py X
__manifest__.py
1  # -*- coding: utf-8 -*-
2  {
3      'name': "managerebeca",
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/technology.xml',
29          'views/project.xml',
30          'views/history.xml',
31          'views/sprint.xml',
32          'views/task.xml',
33          'views/developer.xml',
34          'views/views.xml',
35          'views/templates.xml',
36
37      ],
38      # only loaded in demonstration mode
39      'demo': [
40          'demo/demo.xml',
41      ],
42  }
43
```

En el archivo manifest se deben incluir los enlaces a cada una de las vistas y al archivo de seguridad. Es importante empezar por el archivo de seguridad `ir.model.access.csv` para que todas las vistas puedan cargar correctamente. Después se coloca el enlace a la vista `technology` ya que en ella se declaran los padres de los que descienden el resto de vistas.


```

27     'security/ir.model.access.csv',
28     'views/technology.xml',
29     'views/project.xml',
30     'views/history.xml',
31     'views/sprint.xml',
32     'views/task.xml',
33     'views/developer.xml',
34     'views/views.xml',
35     'views/templates.xml',

```

- Archivo `__init__.py`:

```

__init__.py X
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

```

En el archivo init se deben incluir los enlaces a cada uno de los modelos.

- Archivo `ir.model.access.csv`:

```

ir.model.access.csv X
security > ir.model.access.csv
1  id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2  access_managerebeca_technology,managerebeca.technology,model_managerebeca_technology,base.group_user,1,1,1,1
3  access_managerebeca_project,managerebeca.project,model_managerebeca_project,base.group_user,1,1,1,1
4  access_managerebeca_history,managerebeca.history,model_managerebeca_history,base.group_user,1,1,1,1
5  access_managerebeca_sprint,managerebeca.sprint,model_managerebeca_sprint,base.group_user,1,1,1,1
6  access_managerebeca_task,managerebeca.task,model_managerebeca_task,base.group_user,1,1,1,1

```

El archivo de seguridad que determina los permisos que tiene cada uno de los modelos presentes. Todos los modelos presentes deben estar incluidos en este archivo para funcionar correctamente.

Ampliación del proyecto

Tras la elaboración del proyecto base se añadió a este una ampliación a elección propia. En este caso, el objetivo de dicha ampliación consiste en una mejora de la interfaz visual del módulo, lo que permite obtener información sobre este con un solo vistazo. Con este propósito se han añadido diversos widgets y funciones.

El resultado final de estas modificaciones se podrá ver en el apartado de pruebas de funcionamiento.

A continuación, se muestran las partes del proyecto que han sido modificadas para lograr esta nueva funcionalidad y se procederá a explicar cada una de ellas.

Modelos

- Modelo history.xml:

```
history.py 1 X
models > history.py > ...
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4
5
6  class history(models.Model):
7      _name = 'managerebeca.history'
8      _description = 'managerebeca.history'
9
10     name = fields.Char(String="Nombre")
11     description = fields.Char(String="Descripción")
12     # variables de ampliación
13     progress = fields.Float(String="Progreso", compute="_get_progress") # guarda el progreso de la historia
14     is_favorite = fields.Boolean(default=False) # indica si la historia es favorita
15     # FIN de variables de ampliación
16
17     project_id = fields.Many2one(comodel_name="managerebeca.project", string = "Proyecto", ondelete="set null", inverse_name = "histories_ids")
18     tasks_ids = fields.One2many(string = "Tareas", comodel_name = "managerebeca.task", inverse_name = "history_id")
19     used_technologies = fields.Many2many("managerebeca.technology", compute = "_get_used_technologies")
20
21     def _get_used_technologies(self):
22         for history in self:
23             technologies = None # Array para concatenar todas las tecnologías. Inicialmente no tiene valor
24             for task in history.tasks_ids: # Para cada una de las tareas de la historia
25                 if not technologies:
26                     technologies = task.technologies_ids
27                 else:
28                     technologies = technologies + task.technologies_ids
29             history.used_technologies = technologies # Asignar las tecnologías a la historia
30
31     # funciones de ampliación
32     @api.depends('tasks_ids')
33     def _get_progress(self):
34         # recorre las tareas asociadas a la historia
35         for history in self:
36             if history.tasks_ids:
37                 # suma el progreso de todas las tareas y lo divide entre el número de tareas
38                 history.progress = (sum(task.progress for task in history.tasks_ids) / len(history.tasks_ids))
39             else:
40                 # si no hay tareas asociadas a la historia el progreso es 0
41                 history.progress = 0
42     # FIN de funciones de ampliación
```

A este modelo se ha añadido el campo progress, que sirve para guardar el progreso de la historia. Este campo se autocompleta mediante la función _get_progress que utiliza el campo tasks_ids del modelo. Este método recorre las tareas asociadas a la historia, suma el valor del campo progress de cada tarea y lo divide entre el número de tareas de la historia. El resultado se asigna al campo progress de la historia. En caso de no haber tareas el valor de progress se establece en 0.

```

13 progress = fields.Float(String="Progreso", compute="_get_progress") # guarda el progreso de la historia
14 is_favorite = fields.Boolean(default=False) # indica si la historia es favorita

32 @api.depends('tasks_ids')
33 def _get_progress(self):
34     # recorre las tareas asociadas a la historia
35     for history in self:
36         if history.tasks_ids:
37             # suma el progreso de todas las tareas y lo divide entre el número de tareas
38             history.progress = (sum(task.progress for task in history.tasks_ids) / len(history.tasks_ids))
39         else:
40             # si no hay tareas asociadas a la historia el progreso es 0
41             history.progress = 0

```

También se añadió el campo `is_favorite`, que indica si la historia se ha marcado como favorita.

```

14 is_favorite = fields.Boolean(default=False) # indica si la historia es favorita

```

- Modelo `project.py`:

```

1 # -*- coding: utf-8 -*-
2
3 from odoo import models, fields, api
4
5 class project(models.Model):
6     _name = 'managerebeca.project'
7     _description = 'managerebeca.project'
8
9     name = fields.Char(String="Nombre")
10    description = fields.Char(String="Descripción")
11
12    # variables de ampliación
13    progress = fields.Float(String="Progreso", compute="_get_progress") # guarda el progreso del proyecto
14    is_favorite = fields.Boolean(default=False) # indica si el proyecto es favorita
15    # FIN de variables de ampliación
16
17    histories_ids = fields.One2many("managerebeca.history", string="Historia", inverse_name="project_id")
18    sprints_ids = fields.One2many("managerebeca.sprint", string="Sprint", inverse_name="project_id")
19
20    # funciones de ampliación
21    @api.depends('histories_ids', 'sprints_ids')
22    def _get_progress(self):
23        # recorre las historias y los sprints asociados al proyecto
24        for project in self:
25            if project.histories_ids and project.sprints_ids:
26                # suma el progreso de todas las historias y sprints y lo divide entre el número total de la suma de historias y sprints
27                project.progress = (sum(history.progress for history in project.histories_ids) + sum(sprint.progress for sprint in project.sprints_ids)) / (len(project.histories_ids) + len(project.sprints_ids))
28            # si hay historias pero no hay sprints asociados al proyecto se calcula la suma del progreso de todas las historias y se divide entre el número total de historias
29            elif project.histories_ids and not project.sprints_ids:
30                project.progress = (sum(history.progress for history in project.histories_ids) / len(project.histories_ids))
31            # si hay sprints pero no hay historias asociadas al proyecto se calcula la suma del progreso de todos los sprints y se divide entre el número total de sprints
32            elif project.sprints_ids and not project.histories_ids:
33                project.progress = (sum(sprint.progress for sprint in project.sprints_ids) / len(project.sprints_ids))
34            else:
35                # si no hay tareas asociadas a la historia el progreso es 0
36                project.progress = 0
37    # FIN de funciones de ampliación

```

A este modelo se ha añadido el campo `progress`, que sirve para guardar el progreso del proyecto. Este se autocompleta mediante la función `_get_progress` que utiliza los campos `histories_ids` y `sprints_ids` del modelo.

Este método recorre tanto las historias como los sprints asociados a la historia, suma el valor del campo `progress` de cada historia y sprint y lo divide entre la suma del número de historias y sprints del proyecto. El resultado se asigna al campo `progress` del proyecto. En caso de que haya historias pero no proyectos se sumará el valor del campo `progress` de cada historia y se dividirá entre el número de historias; y, en el caso contrario, que haya sprints pero no historias, se hará lo mismo pero con los sprints en lugar de las historias. Si no hay ni historias ni sprints asociados al proyecto el valor de `progress` se establece en 0.

```

13 progress = fields.Float(String="Progreso", compute="_get_progress") # guarda el progreso del proyecto
14 is_favorite = fields.Boolean(default=False) # indica si la historia es favorita

```

```

21 @api.depends('histories_ids', 'sprints_ids')
22 def _get_progress(self):
23     # recorre las historias y los sprints asociados al proyecto
24     for project in self:
25         if project.histories_ids and project.sprints_ids:
26             # suma el progreso de todas las historias y sprints y lo divide entre el número total de la suma de historias y sprints
27             project.progress = ((sum(history.progress for history in project.histories_ids) + sum(sprint.progress for sprint in project.sprints_ids)) / (len(project.histories_ids) + len(project.sprints_ids)))
28             # si hay historias pero no hay sprints asociados al proyecto se calcula la suma del progreso de todas las historias y se divide entre el número total de historias
29             elif project.histories_ids and not project.sprints_ids:
30                 project.progress = (sum(history.progress for history in project.histories_ids) / len(project.histories_ids))
31             # si hay sprints pero no hay historias asociadas al proyecto se calcula la suma del progreso de todos los sprints y se divide entre el número total de sprints
32             elif project.sprints_ids and not project.histories_ids:
33                 project.progress = (sum(sprint.progress for sprint in project.sprints_ids) / len(project.sprints_ids))
34             else:
35                 # si no hay tareas asociadas a la historia el progreso es 0
36                 project.progress = 0

```

También se añadió el campo `is_favorite`, que indica si el proyecto se ha marcado como favorito.

```

14 is_favorite = fields.Boolean(default=False) # indica si el proyecto es favorita

```

- Modelo `sprint.py`:

```

sprint.py X
models > sprint.py > ...
1 # -*- coding: utf-8 -*-
2
3 from odoo import models, fields, api
4 import datetime
5
6
7 class sprint(models.Model):
8     _name = 'managerebeca.sprint'
9     _description = 'managerebeca.sprint'
10
11     name = fields.Char(string="Nombre")
12     description = fields.Text(string="Descripción")
13     duration = fields.Integer(default = 15)
14     start_date = fields.Datetime(string="Fecha inicio")
15     end_date = fields.Datetime(string="Fecha fin", compute="_get_end_date", store=True)
16     # variables de ampliación
17     progress = fields.Float(string="Progreso", compute="_get_progress") # guarda el progreso del sprint
18     is_favorite = fields.Boolean(default=False) # indica si la tarea es favorita
19     # FIN de variables de ampliación
20
21     tasks_ids = fields.One2many(string = "Tareas", comodel_name = "managerebeca.task", inverse_name = "sprint_id")
22     project_id = fields.Many2one(comodel_name="managerebeca.project", inverse_name="sprints_ids", string="Proyectos")
23
24     @api.depends('start_date', 'duration')
25     def _get_end_date(self):
26         for sprint in self:
27             #try:
28             if isinstance(sprint.start_date, datetime.datetime) and sprint.duration > 0:
29                 sprint.end_date = sprint.start_date + datetime.timedelta(days=sprint.duration)
30             else:
31                 sprint.end_date = sprint.start_date
32
33     # funciones de ampliación
34     @api.depends('tasks_ids')
35     def _get_progress(self):
36         # recorre las tareas asociadas al sprint
37         for sprint in self:
38             if sprint.tasks_ids:
39                 # suma el progreso de todas las tareas y lo divide entre el número de tareas
40                 sprint.progress = (sum(task.progress for task in sprint.tasks_ids) / len(sprint.tasks_ids))
41             else:
42                 # si no hay tareas asociadas al sprint el progreso es 0
43                 sprint.progress = 0
44     # FIN de funciones de ampliación

```

A este modelo se ha añadido el campo `progress`, que sirve para guardar el progreso del sprint. Este campo se autocompleta mediante la función `_get_progress` que utiliza el campo `tasks_ids` del modelo. Este método recorre las tareas asociadas al sprint, suma el valor del campo `progress` de

cada tarea y lo divide entre el número de tareas del sprint. El resultado se asigna al campo progress del sprint. En caso de no haber tareas el valor de progress se establece en 0.

```
17 progress = fields.Float(string="Progreso", compute="_get_progress") # guarda el progreso del sprint
18 is_favorite = fields.Boolean(default=False) # indica si la tarea es favorita

34 @api.depends('tasks_ids')
35 def _get_progress(self):
36     # recorre las tareas asociadas al sprint
37     for sprint in self:
38         if sprint.tasks_ids:
39             # suma el progreso de todas las tareas y lo divide entre el número de tareas
40             sprint.progress = (sum(task.progress for task in sprint.tasks_ids) / len(sprint.tasks_ids))
41         else:
42             # si no hay tareas asociadas al sprint el progreso es 0
43             sprint.progress = 0
```

También se añadió el campo is_favorite, que indica si el sprint se ha marcado como favorito.

```
18 is_favorite = fields.Boolean(default=False) # indica si la tarea es favorita
```

- Modelo task.py:

```
task.py 1 x
models > task.py > ...
1 # -*- coding: utf-8 -*-
2
3 import datetime
4 from odoo import models, fields, api
5
6
7 class task(models.Model):
8     _name = 'managerebeca.task'
9     _description = 'managerebeca.task'
10
11     name = fields.Char(string="Nombre")
12     description = fields.Char(string="Descripción")
13     start_date = fields.Datetime()
14     end_date = fields.Datetime()
15     is_paused = fields.Boolean()
16     # variables de ampliación
17     progress = fields.Float(string="Tiempo transcurrido", compute="_get_progress", store=True) # guarda el progreso de la tarea
18     is_finished = fields.Boolean(string="Está terminada", default=False, store=True) # indica si la tarea ha terminado
19     is_favorite = fields.Boolean(default=False) # indica si la tarea es favorita
20     state = fields.Selection([
21         ('paused', 'En pausa'),
22         ('finished', 'Finalizada'),
23         ('in_progress', 'En progreso')
24     ], default='in_progress', string='Estado', compute="_change_state") # define el estado de la tarea
25     # FIN de variables de ampliación
26
27     sprint_id = fields.Many2one(comodel_name="managerebeca.sprint", compute = "_get_sprint", store = True, inverse_name="tasks_ids")
28     technologies_ids = fields.Many2many(comodel_name = "managerebeca.technology",
29         relation = "technologies_tasks",
30         column1 = "technologies_ids",
31         column2 = "tasks_ids",
32         string = "Tecnologías",)
33     history_id = fields.Many2one(comodel_name="managerebeca.history", ondelete="set null", help="Historia relacionada", inverse_name = "tasks_ids")
34     definition_date = fields.Datetime(default=lambda p: datetime.datetime.now())
35     project_id = fields.Many2one(comodel_name="managerebeca.project", related = "history_id.project_id", readonly = True)
36     code = fields.Char(string = "Código", compute = "_get_code")
37
38     @api.depends('start_date', 'end_date')
39     def _get_sprint(self):
40         for task in self:
41             sprints = self.env['managerebeca.sprint'].search([('project_id', '=', task.history_id.project_id.id)])
42             found = False
43             for sprint in sprints:
44                 if isinstance(sprint.end_date, datetime.datetime) and sprint.end_date > datetime.datetime.now():
45                     task.sprint_id = sprint.id
46                     found = True
47             if not found:
48                 task.sprint_id = False
```

```

48         task.sprint_id = False
49
50     #@api.one
51     def _get_code(self):
52         for task in self:
53             # try:
54             task.code = "TSK_"+str(task.id)
55             # logger.info("Código generado: "+task.code)
56             #except:
57             #raise ValidationError(_("Generación de código errónea"))
58
59     # funciones de ampliación
60     @api.depends('start_date', 'end_date', 'is_finished')
61     def _get_progress(self):
62         for task in self:
63             if task.start_date and task.end_date:
64                 # calcula el tiempo total disponible para completar la tarea
65                 total_time = (task.end_date - task.start_date).total_seconds()
66                 # calcula el tiempo transcurrido hasta la fecha actual
67                 elapsed_time = (datetime.datetime.now() - task.start_date).total_seconds()
68                 if total_time > 0:
69                     if task.is_finished == False:
70                         # calcula el porcentaje del tiempo transcurrido respecto al tiempo total
71                         progress = (elapsed_time / total_time) * 100
72                         if progress >= 100:
73                             # impide que el porcentaje de avance supere el 100%
74                             task.progress = 100
75                         else:
76                             task.progress = progress
77                         # si la tarea ha sido marcada como finalizada el porcentaje se establece como 100%
78                         else: task.progress = 100
79                         # si el tiempo total es menor que 0 el progreso es 0
80                         else:
81                             task.progress = 0
82                         # si la tarea no tiene fecha de inicio o fecha de finalización el progreso es 0
83                         else:
84                             task.progress = 0
85
86     # función para el botón FINALIZAR
87     def f_finish_task(self):
88         for task in self:
89             task.is_finished = True
90
91     # función para el botón CONTINUAR
92     def f_continue_task(self):
93         for task in self:
94             task.is_finished = False
95
96     # cambia el estado de la tarea
97     @api.depends('is_paused', 'progress')
98     def _change_state(self):
99         for task in self:
100             if task.progress == 100:
101                 task.state = 'finished'
102             elif task.is_paused:
103                 task.state = 'paused'
104             else:
105                 task.state = 'in_progress'
106     # FIN de funciones de ampliación

```

A este modelo se ha añadido el campo progress, que sirve para guardar el progreso de la tarea. Este campo se autocompleta mediante la función _get_progress que utiliza los campos start_date, end_date y is_finished del modelo.

Este método lee el valor de la fecha de inicio y finalización de la tarea y las resta para obtener la duración total de la misma. Luego resta la fecha de inicio a la fecha actual para calcular el tiempo transcurrido desde que comenzó la tarea. Por último calcula el porcentaje que representa el tiempo transcurrido frente al tiempo total. El resultado se asigna al campo progress de la tarea. Si este valor supera 100, el valor se establece en 100 como valor fijo.

Por otro lado, si la tarea se ha establecido como finalizada mediante los botones que explicaremos más adelante, el valor también se establecerá como 100. En caso de que el tiempo sea 0 o inferior

el progreso se establece en 0 para evitar errores, al igual que si la fecha de inicio o de finalización son 0, el valor del progreso también será 0.

```
17 progress = fields.Float(String="Tiempo transcurrido", compute="_compute_progress", store=True) # guarda el progreso de la tarea

60 @api.depends('start_date', 'end_date', 'is_finished')
61 def _get_progress(self):
62     for task in self:
63         if task.start_date and task.end_date:
64             # calcula el tiempo total disponible para completar la tarea
65             total_time = (task.end_date - task.start_date).total_seconds()
66             # calcula el tiempo transcurrido hasta la fecha actual
67             elapsed_time = (datetime.datetime.now() - task.start_date).total_seconds()
68             if total_time > 0:
69                 if task.is_finished == False:
70                     # calcula el porcentaje del tiempo transcurrido respecto al tiempo total
71                     progress = (elapsed_time / total_time) * 100
72                     if progress >= 100:
73                         # impide que el porcentaje de avance supere el 100%
74                         task.progress = 100
75                     else:
76                         task.progress = progress
77                     # si la tarea ha sido marcada como finalizada el porcentaje se establece como 100%
78                     else: task.progress = 100
79                     # si el tiempo total es menor que 0 el progreso es 0
80                     else:
81                         task.progress = 0
82                     # si la tarea no tiene fecha de inicio o fecha de finalización el progreso es 0
83                     else:
84                         task.progress = 0
```

También se añadió el campo `is_finished`, que indica si la tarea se ha establecido como finalizada.

```
18 is_finished = fields.Boolean(String="Está terminada", default=False, store=True) # indica si la tarea ha terminado
```

Además, se añadió el campo `is_favorite`, que indica si la tarea se ha marcado como favorita.

```
19 is_favorite = fields.Boolean(default=False) # indica si la tarea es favorita
```

Por último, el campo `state` define el estado en que se encuentra la tarea para modificar el color de su insignia. Estos estados pueden ser pausado, finalizado o en progreso. Este campo se calcula automáticamente mediante el método `_change_state` que utiliza los campos `is_paused` y `progress`.

Este método comprueba distintos campos del modelo `task` para cambiar el estado. Si el campo `progress` de la tarea es 100 la tarea se establecerá como `finished`. Por otro lado, si el campo `is_paused` está como verdadero la tarea se establecerá como `paused`. En cualquier otro caso la tarea se establecerá como `in_progress`.

```
20 state = fields.Selection([
21     ('paused', 'En pausa'),
22     ('finished', 'Finalizada'),
23     ('in_progress', 'En progreso')
24 ], default='in_progress', string='Estado', compute="_change_state") # define el estado de la tarea
```

```

96     # cambia el estado de la tarea
97     @api.depends('is_paused', 'progress')
98     def _change_state(self):
99         for task in self:
100             if task.progress == 100:
101                 task.state = 'finished'
102             elif task.is_paused:
103                 task.state = 'paused'
104             else:
105                 task.state = 'in_progress'

```

Este modelo también cuenta con la función `f_finish_task`, que es utilizada por el botón FINALIZAR de la vista para establecer el campo `is_finished` a verdadero.

```

86     # función para el botón FINALIZAR
87     def f_finish_task(self):
88         for task in self:
89             task.is_finished = True

```

Por el contrario la función `f_continue_task` establece el campo `is_finished` a falso al pulsar el botón CONTINUAR de la vista.

```

91     # función para el botón CONTINUAR
92     def f_continue_task(self):
93         for task in self:
94             task.is_finished = False

```

- Modelo `technology.py`:

```

technology.py 1 X
models > technology.py > ...
1  # -*- coding: utf-8 -*-
2
3  from odoo import models, fields, api
4
5
6  class technology(models.Model):
7      _name = 'managerebeca.technology'
8      _description = 'managerebeca.technology'
9
10     name = fields.Char(String="Nombre")
11     description = fields.Char(String="Descripción")
12     photo = fields.Image(string="Imagen", max_width=200, max_height=200)
13     # variables de ampliación
14     is_favorite = fields.Boolean(default=False) # indica si la tecnología es favorita
15     # FIN de variables de ampliación
16
17     tasks_ids = fields.Many2many(comodel_name = "managerebeca.task",
18                                relation = "technologies_tasks",
19                                column1 = "tasks_ids",
20                                column2 = "technologies_ids")

```


Se añadió el campo `is_favorite`, que indica si la tecnología se ha marcado como favorita.

```
14     is_favorite = fields.Boolean(default=False) # indica si la tecnología es favorita
```

Vistas

- Vista history.xml:

```

history.xml x
views > history.xml > odoo > data > record > field > form
1 <odoo>
2 <data>
3 <!-- explicit list view definition -->
4 <record model="ir.ui.view" id="vista_managerebeca_history_tree">
5 <field name="name">vista_managerebeca_history_tree</field>
6 <field name="model">managerebeca.history</field>
7 <field name="arch" type="xml">
8 <tree>
9 <!-- campos de ampliación -->
10 <field name="is_favorite" widget="boolean_favorite" nolabel="1" />
11 <!-- FIN de campos de ampliación -->
12 <field name="name" />
13 <field name="description" />
14 <!-- campos de ampliación -->
15 <field name="progress" widget="progressbar" />
16 <!-- FIN de campos de ampliación -->
17 </tree>
18 </field>
19 </record>
20
21 <record model="ir.ui.view" id="vista_managerebeca_history_form">
22 <field name="name">vista_managerebeca_history_form</field>
23 <field name="model">managerebeca.history</field>
24 <field name="arch" type="xml">
25 <form string="formulario_history">
26 <sheet>
27 <group name="group_top">
28 <field name="name" />
29 <field name="description" />
30 <field name="project_id" />
31 <field name="tasks_ids" />
32 <field name="used_technologies" />
33 </group>
34 </sheet>
35 </form>
36 </field>
37 </record>
38
39 <record model="ir.actions.act_window" id="accion_managerebeca_history_form">
40 <field name="name">Listado de historias</field>
41 <field name="type">ir.actions.act_window</field>
42 <field name="res_model">managerebeca.history</field>
43 <field name="view_mode">tree,form</field>
44 <field name="help" type="html">
45 <p class="oe_view_nocontent_create">
46 Historial
47 </p>
48 <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
49 </p>
50 </field>
51 </record>
52
53 <!-- actions -->
54
55 <menuitem name="Histories" id="menu_managerebeca_history" parent="menu_managerebeca_management"
56 action="accion_managerebeca_history_form" />
57
58 </data>
59 </odoo>

```

A esta vista le hemos añadido el campo `is_favorite`. Este es un widget de tipo `boolean_favorite`, que se ve como una pequeña estrella sobre la que puedes hacer clic para que se vuelva amarilla, y volver a pulsar para que vuelva a la normalidad. Esta acción cambia el valor de `is_favorite` entre verdadero y falso respectivamente.

```
10 <field name="is_favorite" widget="boolean_favorite" nolabel="1" />
```

También se añadió el campo `progress`. Este es un widget de tipo `progressbar`, que muestra una barra que se va rellenando dependiendo del progreso del campo `progress` del modelo `history.py`, junto con su respectivo porcentaje al lado.

```
15 <field name="progress" widget="progressbar" />
```

- Vista project.xml:

```

1 <!-- explicit list view definition -->
2 <record model="ir.ui.view" id="vista_managerebeca_project_tree">
3   <field name="name">vista_managerebeca_project_tree</field>
4   <field name="model">managerebeca.project</field>
5   <field name="arch" type="xml">
6     <tree>
7       <!-- campos de ampliación -->
8       <field name="is_favorite" widget="boolean_favorite" nolabel="1" />
9       <!-- FIN de campos de ampliación -->
10      <field name="name" />
11      <field name="description" />
12      <!-- campos de ampliación -->
13      <field name="progress" widget="progressbar" />
14      <!-- FIN de campos de ampliación -->
15    </tree>
16  </field>
17 </record>
18
19 <record model="ir.ui.view" id="vista_managerebeca_project_form">
20   <field name="name">vista_managerebeca_project_form</field>
21   <field name="model">managerebeca.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           <field name="histories_ids" />
29           <field name="sprints_ids" />
30         </group>
31       </sheet>
32     </form>
33   </field>
34 </record>
35
36 <record model="ir.actions.act_window" id="accion_managerebeca_project_form">
37   <field name="name">Listado de proyectos</field>
38   <field name="type">ir.actions.act_window</field>
39   <field name="res_model">managerebeca.project</field>
40   <field name="view_mode">tree,form</field>
41   <field name="help" type="html">
42     <p class="oe_view_nocontent_create">
43       Listado de proyectos
44     </p>
45     <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
46   </p>
47 </field>
48 </record>
49
50 <!-- actions -->
51
52 <menuitem name="Projects" id="menu_managerebeca_project" parent="menu_managerebeca_management"
53   action="accion_managerebeca_project_form" />
54
55 </data>
56 </odoo>

```

A esta vista le hemos añadido el campo `is_favorite`. Este es un widget de tipo `boolean_favorite`, que se ve como una pequeña estrella sobre la que puedes hacer clic para que se vuelva amarilla, y volver a pulsar para que vuelva a la normalidad. Esta acción cambia el valor de `is_favorite` entre verdadero y falso respectivamente.

```
10 <field name="is_favorite" widget="boolean_favorite" nolabel="1" />
```

También se añadió el campo progress. Este es un widget de tipo progressbar, que muestra una barra que se va rellenando dependiendo del progreso del campo progress del modelo history.py, junto con su respectivo porcentaje al lado.

```
15 <field name="progress" widget="progressbar" />
```

- Vista sprint.xml:

```
sprint.xml x
views > sprint.xml > odoo
1 <odoo>
2   <data>
3     <!-- explicit list view definition -->
4     <record model="ir.ui.view" id="vista_managerebeca_sprint_tree">
5       <field name="name">vista_managerebeca_sprint_tree</field>
6       <field name="model">managerebeca.sprint</field>
7       <field name="arch" type="xml">
8         <tree>
9           <!-- campos de ampliación -->
10          <field name="is_favorite" widget="boolean_favorite" nolabel="1" />
11          <!-- FIN de campos de ampliación -->
12          <field name="name" />
13          <field name="description" />
14          <!-- campos de ampliación -->
15          <field name="progress" widget="progressbar" />
16          <!-- FIN de campos de ampliación -->
17        </tree>
18      </field>
19    </record>
20
21    <record model="ir.ui.view" id="vista_managerebeca_sprint_form">
22      <field name="name">vista_managerebeca_sprint_form</field>
23      <field name="model">managerebeca.sprint</field>
24      <field name="arch" type="xml">
25        <form string="formulario_sprint">
26          <sheet>
27            <group name="group_top">
28              <field name="name" />
29              <field name="description" />
30              <field name="duration" />
31              <field name="start_date" />
32              <field name="end_date" />
33              <field name="tasks_ids" />
34              <field name="project_id" />
35            </group>
36          </sheet>
37        </form>
38      </field>
39    </record>
40
41    <record model="ir.actions.act_window" id="accion_managerebeca_sprint_form">
42      <field name="name">Listado de sprints</field>
43      <field name="type">ir.actions.act_window</field>
44      <field name="res_model">managerebeca.sprint</field>
45      <field name="view_mode">tree,form</field>
46      <field name="help" type="html">
47        <p class="oe_view_nocontent_create">
48          Listado de sprints
```

```

49         </p>
50         <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
51         </p>
52     </field>
53 </record>
54
55 <!-- actions -->
56
57     <menuitem name="Sprints" id="menu_managerebeca_sprint" parent="menu_managerebeca_management"
58         action="accion_managerebeca_sprint_form" />
59
60 </data>
61 </odoo>

```

A esta vista le hemos añadido el campo `is_favorite`. Este es un widget de tipo `boolean_favorite`, que se ve como una pequeña estrella sobre la que puedes hacer clic para que se vuelva amarilla, y volver a pulsar para que vuelva a la normalidad. Esta acción cambia el valor de `is_favorite` entre verdadero y falso respectivamente.

```

10         <field name="is_favorite" widget="boolean_favorite" nolabel="1" />

```

También se añadió el campo `progress`. Este es un widget de tipo `progressbar`, que muestra una barra que se va rellenando dependiendo del progreso del campo `progress` del modelo `history.py`, junto con su respectivo porcentaje al lado.

```

15         <field name="progress" widget="progressbar" />

```

- Vista task.xml:

```
task.xml x
views > task.xml > odoo
1 <odoo>
2   <data>
3     <!-- explicit list view definition -->
4     <record model="ir.ui.view" id="vista_managerebeca_task_tree">
5       <field name="name">vista_managerebeca_task_tree</field>
6       <field name="model">managerebeca.task</field>
7       <field name="arch" type="xml">
8         <tree>
9           <!-- campos de ampliación -->
10          <field name="is_favorite" widget="boolean_favorite" nolabel="1" />
11          <!-- FIN de campos de ampliación -->
12          <field name="name" />
13          <field name="description" />
14          <!-- campos campos de ampliación -->
15          <field name="progress" widget="progressbar" />
16          <button name="f_finish_task" string="Finalizar" class="oe_highlight" type="object" />
17          <button name="f_continue_task" string="Continuar" class="oe_highlight" type="object" />
18          <field name="state">
19            decoration-warning="state == 'paused'"
20            decoration-success="state == 'finished'"
21            decoration-muted="state == 'in_progress'"
22            widget="badge" >/field>
23          <!-- FIN de campos campos de ampliación -->
24        </tree>
25      </field>
26    </record>
27
28    <record model="ir.ui.view" id="vista_managerebeca_task_form">
29      <field name="name">vista_managerebeca_task_form</field>
30      <field name="model">managerebeca.task</field>
31      <field name="arch" type="xml">
32        <form string="formulario_task">
33          <sheet>
34            <group name="group_top">
35              <field name="code" />
36              <field name="name" />
37              <field name="description" />
38              <field name="start_date" />
39              <field name="end_date" />
40              <field name="is_paused" />
41              <field name="sprint_id" />
42              <field name="history_id" />
43              <field name="technologies_ids" />
44              <field name="definition_date" />
45            </group>
46          </sheet>
47        </form>
48      </field>
```

```

49     </record>
50
51     <record model="ir.actions.act_window" id="accion_managerebeca_task_form">
52         <field name="name">Listado de tareas</field>
53         <field name="type">ir.actions.act_window</field>
54         <field name="res_model">managerebeca.task</field>
55         <field name="view_mode">tree,form</field>
56         <field name="help" type="html">
57             <p class="oe_view_nocontent_create">
58                 Listado de tareas
59             </p>
60             <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
61             </p>
62         </field>
63     </record>
64
65     <!-- actions -->
66
67     <menuitem name="Tasks" id="menu_managerebeca_task" parent="menu_managerebeca_management"
68         action="accion_managerebeca_task_form" />
69
70 </data>
71 </odoo>

```

A esta vista le hemos añadido el campo `is_favorite`. Este es un widget de tipo `boolean_favorite`, que se ve como una pequeña estrella sobre la que puedes hacer clic para que se vuelva amarilla, y volver a pulsar para que vuelva a la normalidad. Esta acción cambia el valor de `is_favorite` entre verdadero y falso respectivamente.

```

10     <field name="is_favorite" widget="boolean_favorite" nolabel="1" />

```

También se añadió el campo `progress`. Este es un widget de tipo `progressbar`, que muestra una barra que se va rellenando dependiendo del progreso del campo `progress` del modelo `history.py`, junto con su respectivo porcentaje al lado.

```

15     <field name="progress" widget="progressbar" />

```

En la vista de las tareas se incluyen 2 botones, el botón FINALIZAR para finalizar las tareas mediante el método `f_finish_task` y el botón CONTINUAR para eliminar el estado finalizado de una tarea mediante el método `f_continue_task`.

```

16     <button name="f_finish_task" string="Finalizar" class="oe_highlight" type="object" />

```

```

17     <button name="f_continue_task" string="Continuar" class="oe_highlight" type="object" />

```

Por último hay una etiqueta que cambia de color dependiendo del valor del campo `state`.

```

18     <field name="state"
19         decoration-warning="state == 'paused'"
20         decoration-success="state == 'finished'"
21         decoration-muted="state == 'in_progress'"
22         widget="badge" ></field>

```


- Vista technology.xml:

```

1 <!-- technology.xml -->
2 <data>
3   <!-- explicit list view definition -->
4   <record model="ir.ui.view" id="vista_managerebeca_technology_tree">
5     <field name="name">vista_managerebeca_technology_tree</field>
6     <field name="model">managerebeca.technology</field>
7     <field name="arch" type="xml">
8       <tree>
9         <!-- campos de ampliación -->
10        <field name="is_favorite" widget="boolean_favorite" nolabel="1" />
11        <!-- FIN de campos de ampliación -->
12        <field name="name" />
13        <field name="description" />
14      </tree>
15    </field>
16  </record>
17
18  <record model="ir.ui.view" id="vista_managerebeca_technology_form">
19    <field name="name">vista_managerebeca_technology_form</field>
20    <field name="model">managerebeca.technology</field>
21    <field name="arch" type="xml">
22      <form string="formulario_technology">
23        <sheet>
24          <group name="group_top">
25            <field name="name" />
26            <field name="description" />
27            <field name="photo" />
28            <field name="tasks_ids" />
29          </group>
30        </sheet>
31      </form>
32    </field>
33  </record>
34
35  <record model="ir.actions.act_window" id="accion_managerebeca_technology_form">
36    <field name="name">Listado de tecnologías</field>
37    <field name="type">ir.actions.act_window</field>
38    <field name="res_model">managerebeca.technology</field>
39    <field name="view_mode">tree,form</field>
40    <field name="help" type="html">
41      <p class="oe_view_nocontent_create">
42        Listado de tecnologías
43      </p>
44      <p>Click <strong> 'Crear' </strong> para añadir nuevos elementos
45      </p>
46    </field>
47  </record>
48
49  <!-- Top menu item -->
50
51  <menuitem name="Manage de Rebeca" id="menu_managerebeca_raiz" />
52
53  <!-- menu categories -->
54
55  <menuitem name="Management" id="menu_managerebeca_management" parent="menu_managerebeca_raiz" />
56
57  <!-- actions -->
58
59  <menuitem name="Technologies" id="menu_managerebeca_technology" parent="menu_managerebeca_management"
60    action="accion_managerebeca_technology_form" />
61
62 </data>
63 </odoo>

```

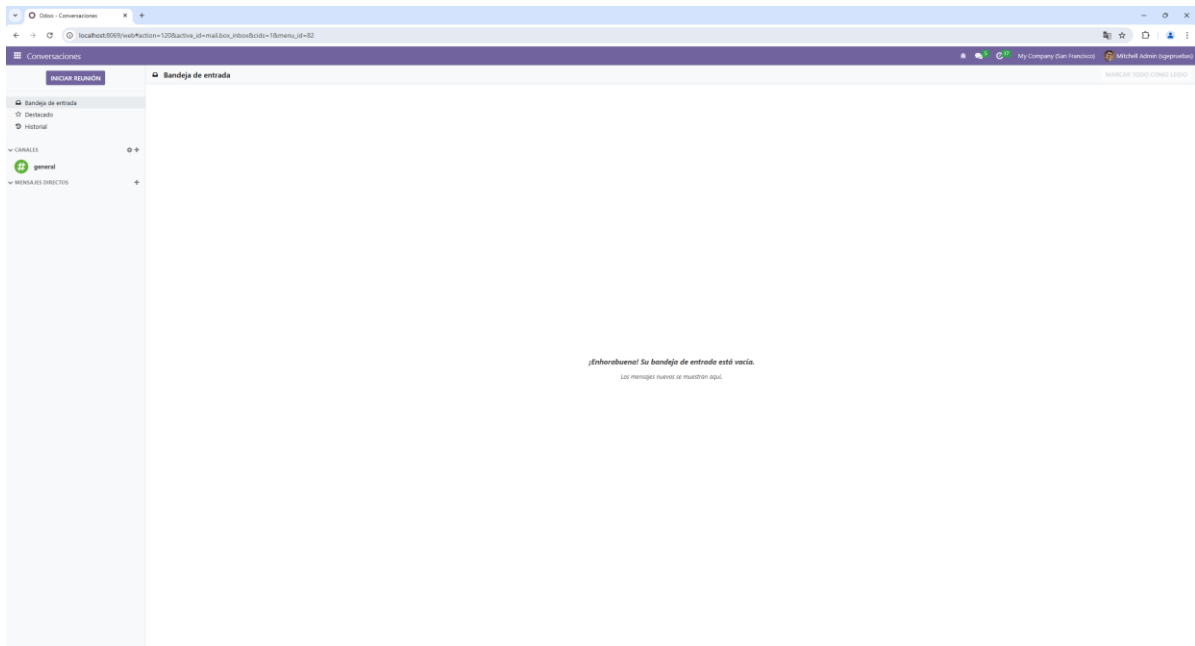
A esta vista le hemos añadido el campo `is_favorite`. Este es un widget de tipo `boolean_favorite`, que se ve como una pequeña estrella sobre la que puedes hacer clic para que se vuelva amarilla, y volver a pulsar para que vuelva a la normalidad. Esta acción cambia el valor de `is_favorite` entre verdadero y falso respectivamente.

```
10 <field name="is_favorite" widget="boolean_favorite" nolabel="1" />
```

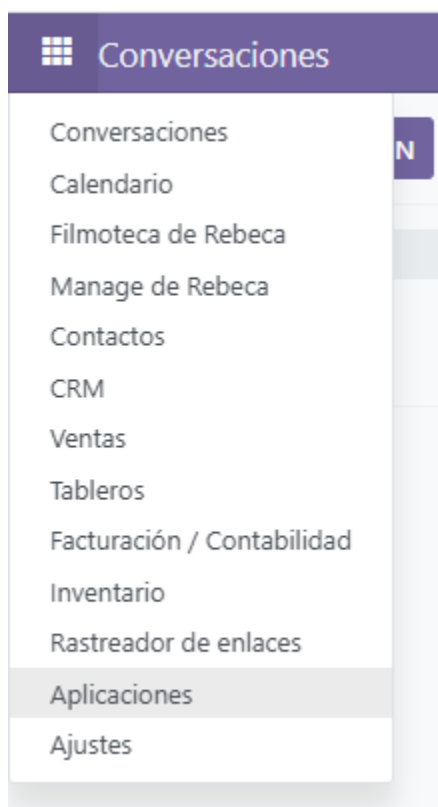
Pruebas de funcionamiento

A continuación, demostraremos el funcionamiento del módulo mediante el uso de ciertas pruebas.

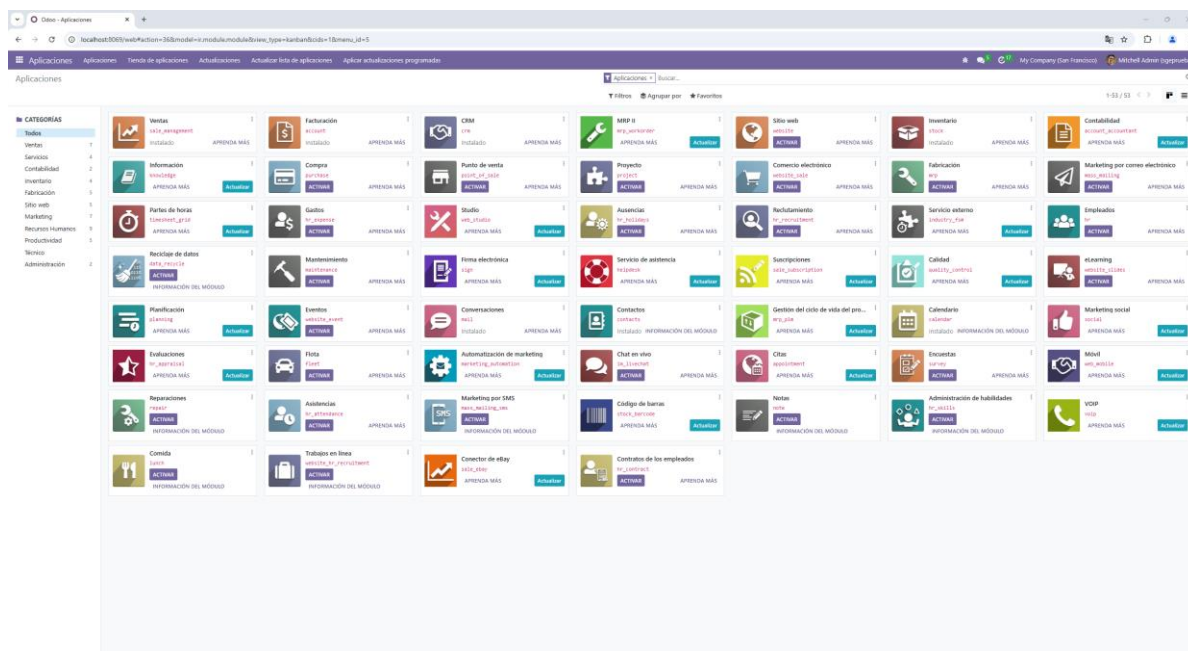
Para empezar, ejecutaremos Docker como modo administrador e iniciaremos la ejecución de los contenedores de Odoo. En nuestro navegador escogido buscaremos `localhost:8069`. Es posible que tengamos que iniciar sesión con nuestras credenciales de Odoo.

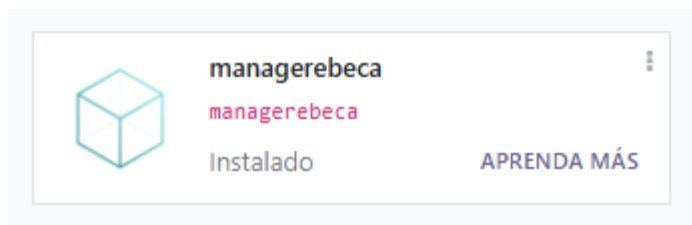
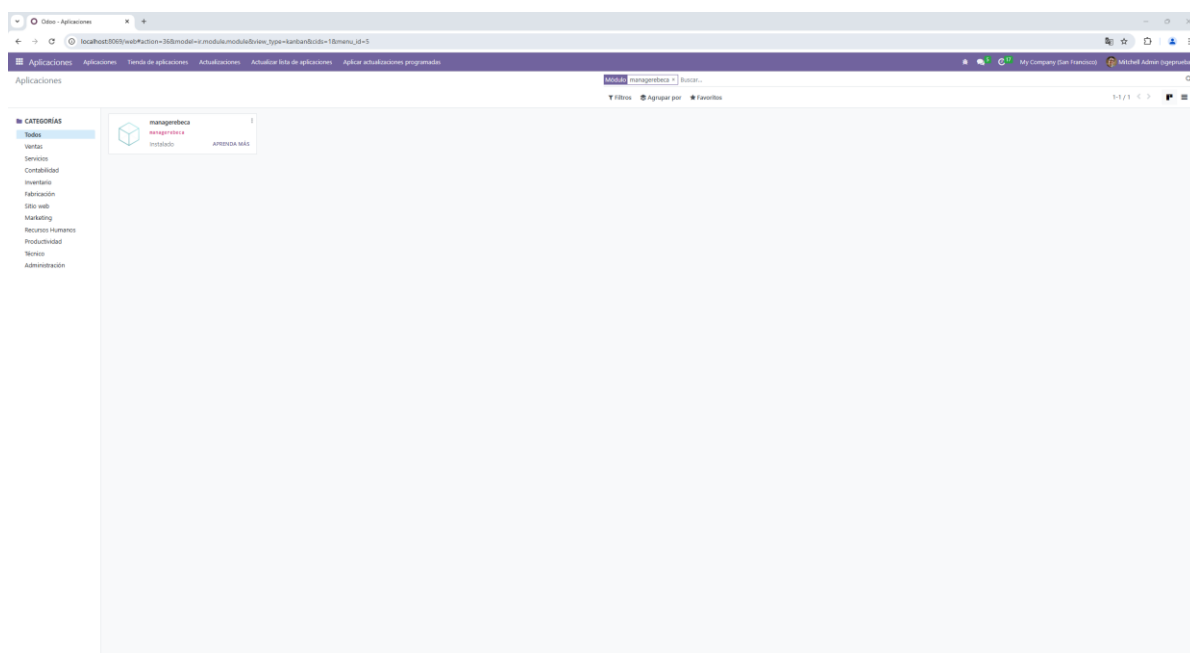
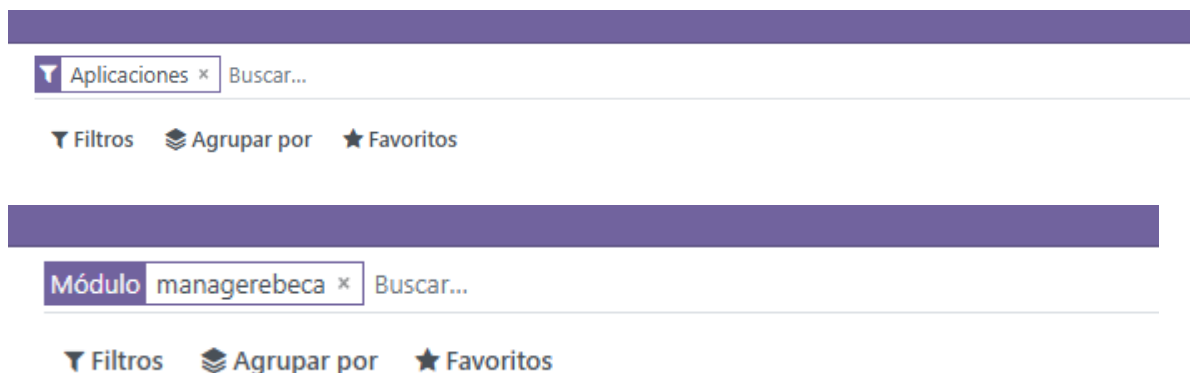
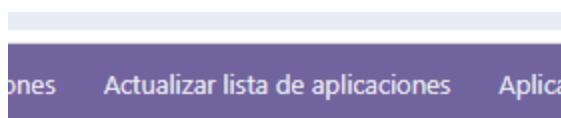


Una vez en el sistema de Odoo debemos dirigirnos al menú de la parte superior izquierda y seleccionar la opción Aplicaciones.

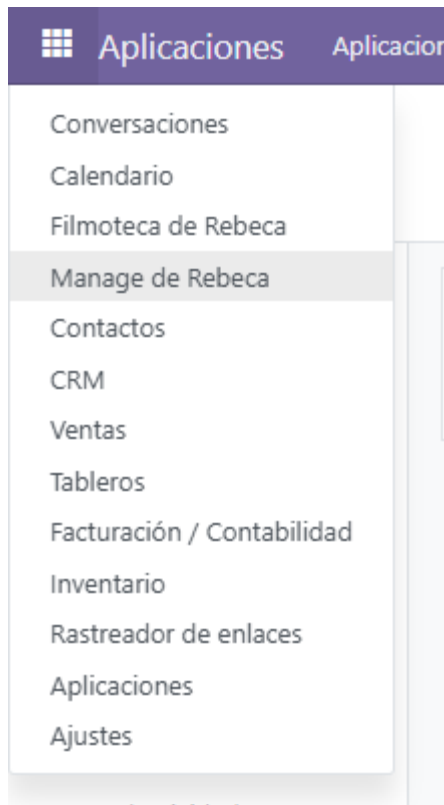


Aquí pulsamos la opción Actualizar lista de aplicaciones de la barra superior. Después en el buscador borramos el filtro por defecto y buscamos el nombre de nuestra aplicación, en este caso managerebeca. Instalamos el módulo.

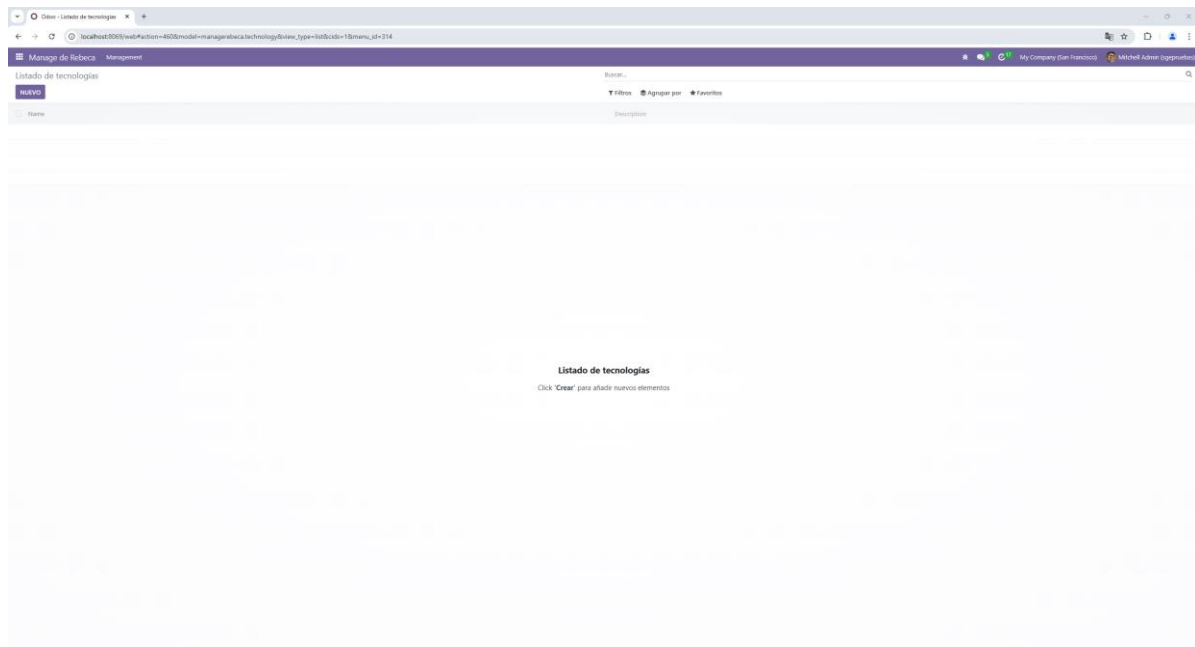




Ahora nuestro módulo aparecerá en el menú superior izquierdo y podremos acceder a él. Desplegamos el menú y accedemos a Manage de Rebeca.



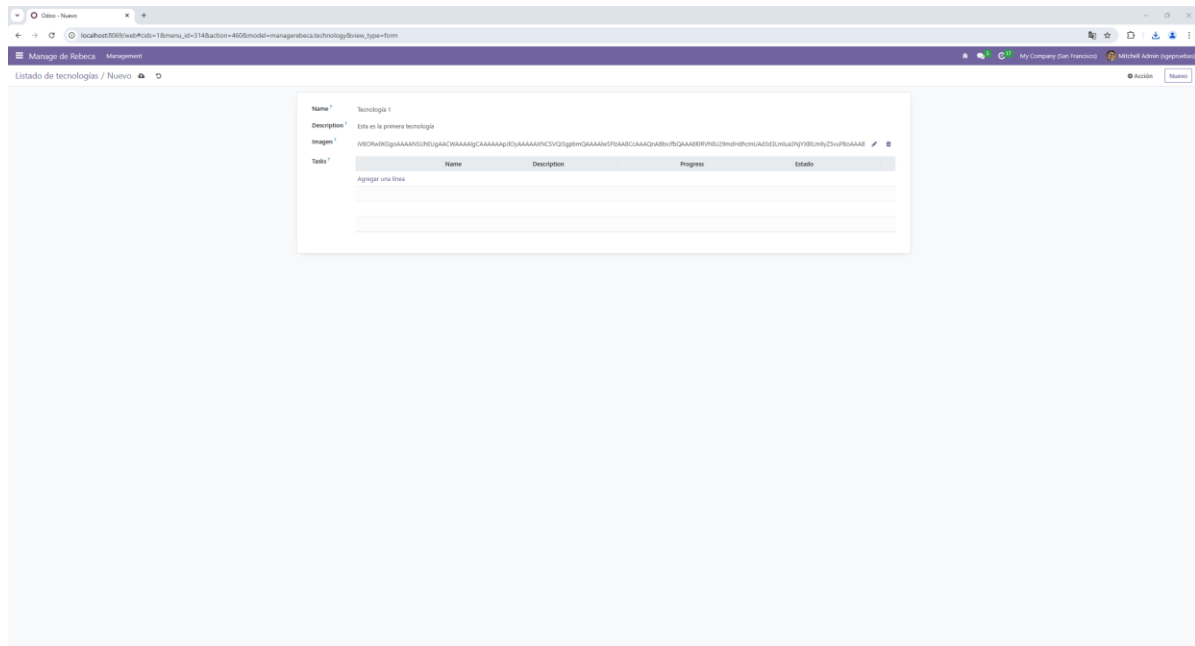
Al iniciar el módulo, lo primero que vemos es la lista de tecnologías. Vamos a crear un par de tecnologías para utilizarlas después. Para ello pulsamos el botón NUEVO.



LISTADO de tecn

NUEVO

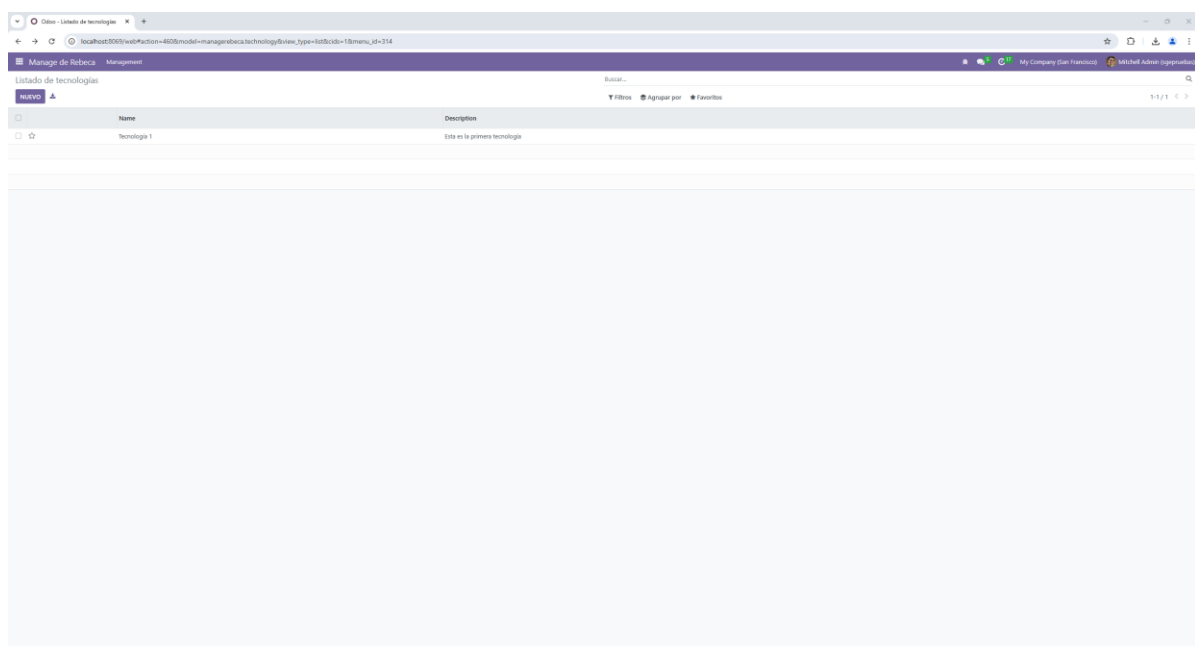
Rellenamos los datos y pulsamos en Listado de tecnologías para volver a la vista de la lista.



Manage de Rebecca

Listado de tecnologías / Tec

Podemos ver que se ha añadido una nueva fila a la lista con la información de la tecnología que acabamos de incluir. Vamos a añadir un par más.




	Name	Description
<input type="checkbox"/>	Tecnología 1	Esta es la primera tecnología

No es necesario volver a la lista cada vez que queramos crear una nueva entrada. En la interfaz donde se rellenan los datos podemos encontrar el botón NUEVO en la esquina superior derecha.

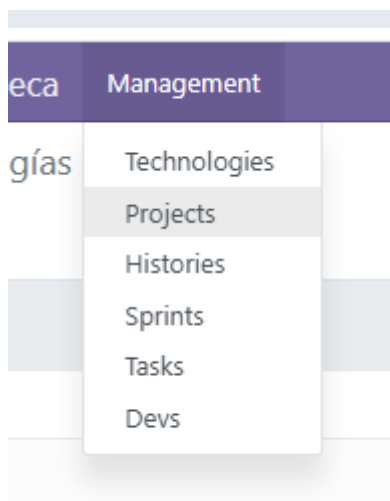


Si pulsamos sobre la estrella de cada tecnología podemos marcarlas como favoritas.

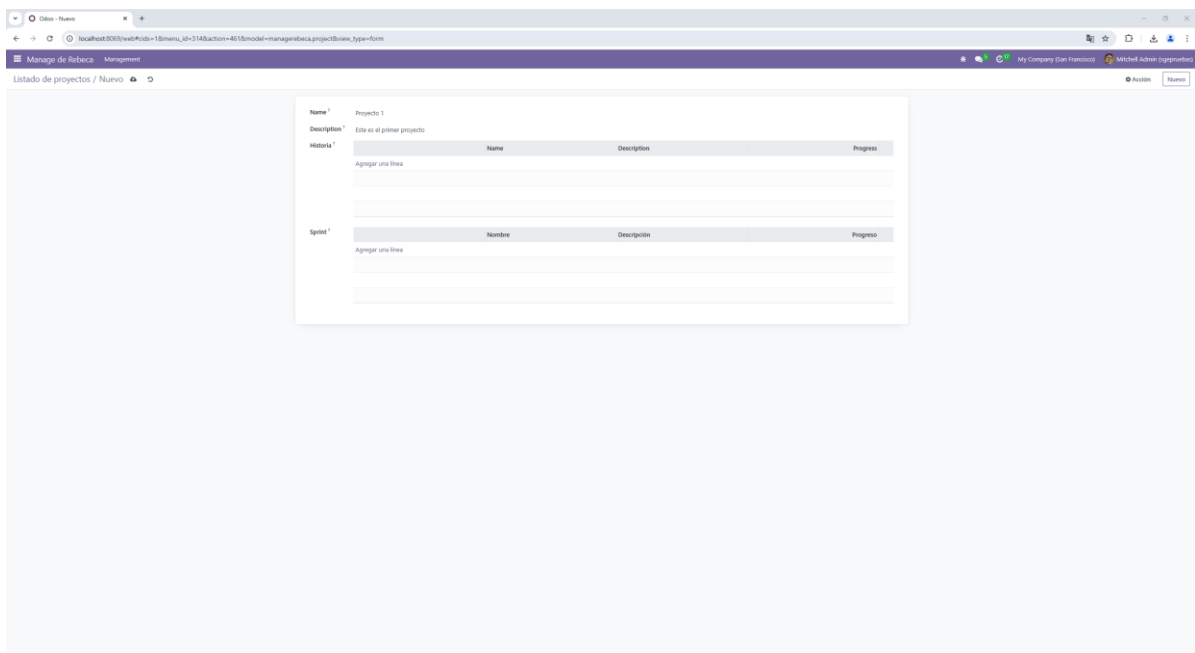
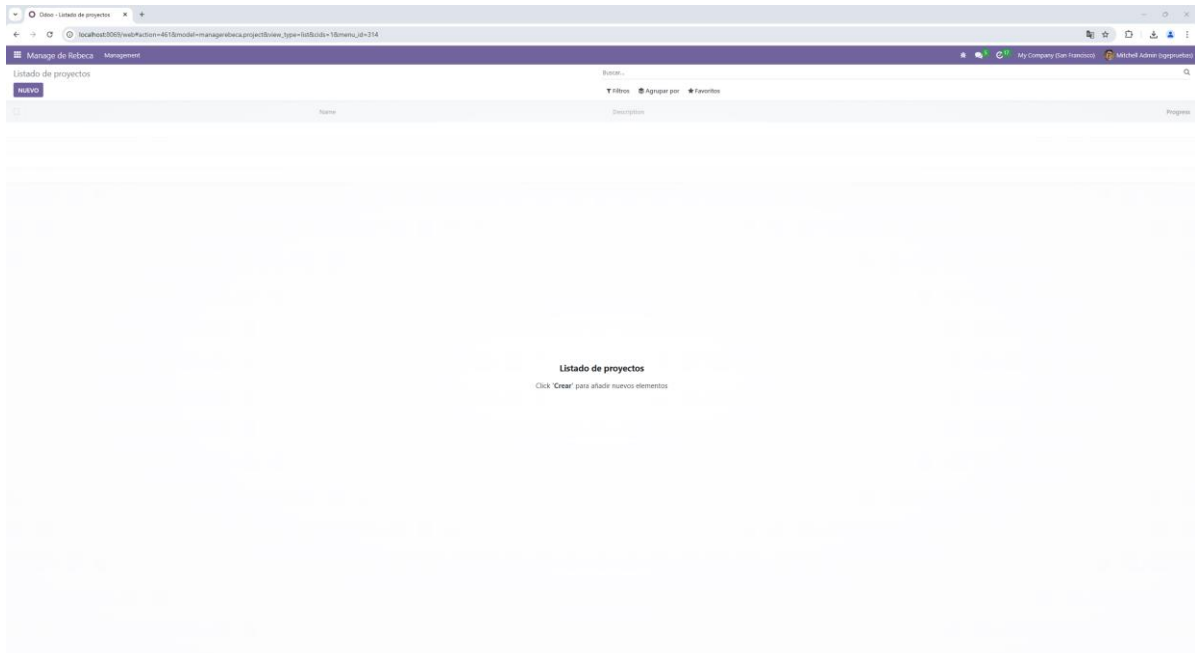
NUEVO 

<input type="checkbox"/>		Name	Description
<input type="checkbox"/>	☆	Tecnología 1	Esta es la primera tecnología
<input type="checkbox"/>	★	Tecnología 2	Esta es la segunda tecnología
<input type="checkbox"/>	☆	Tecnología 3	Esta es la tercera tecnología

Ahora vamos a crear nuestro proyecto. Para ello nos dirigimos a la barra superior y pulsamos sobre Management. Esto despliega el menú de todos los apartados del módulo. Vamos al apartado Projects.



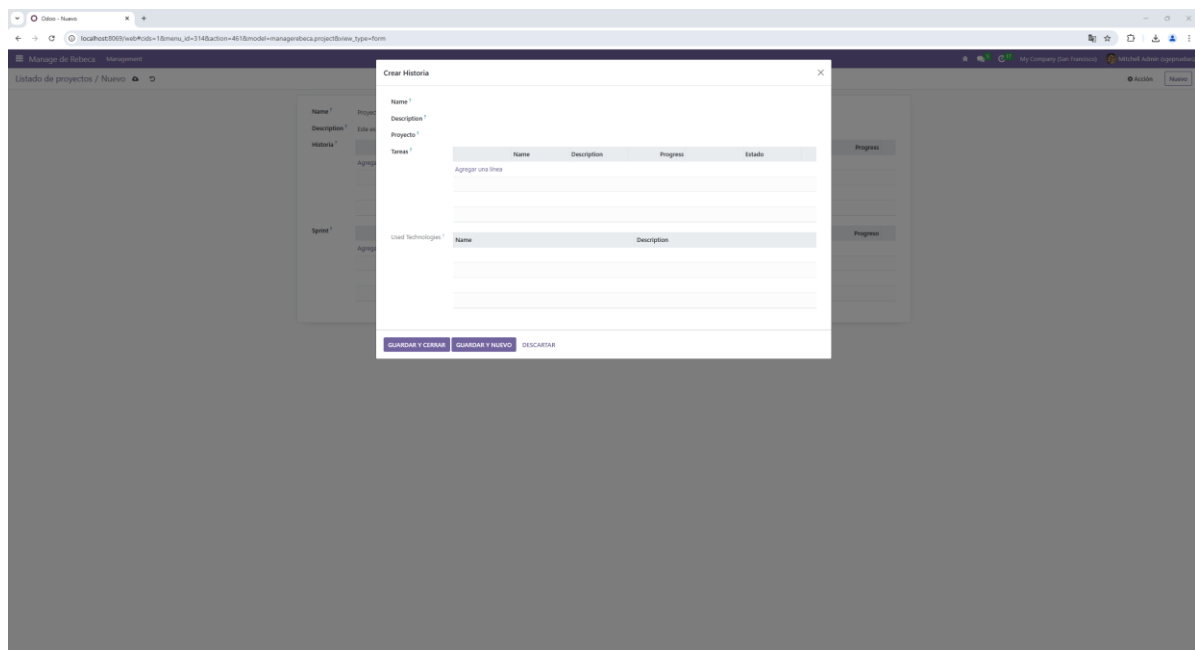
Nos sale un menú igual que el de tecnologías. Pulsamos en NUEVO para crear el proyecto y rellenamos los datos requeridos.



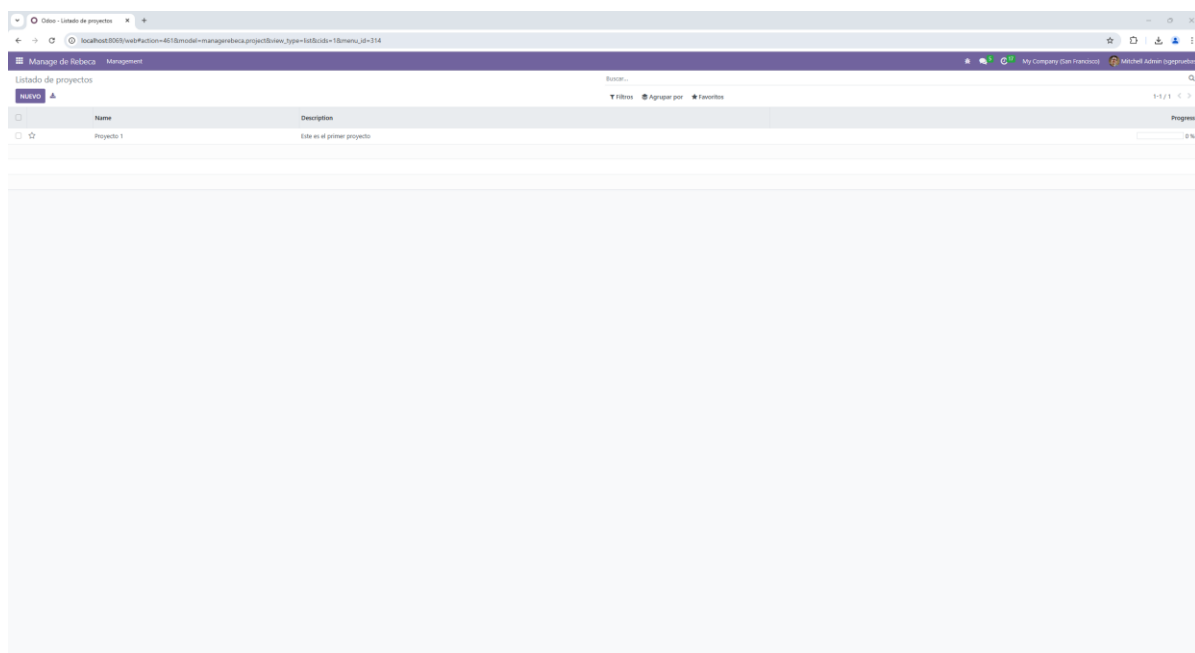
Desde estos menús también se puede acceder al menú de creación de otras partes del módulo, en este caso al menú de creación de historias y sprints. Para mantener la claridad y linealidad de la explicación, no crearemos nada de esta forma, pero es importante notar que podría hacerse fuese este el deseo del usuario. Para ello solo habría que pulsar Agregar una línea en cualquiera de estas tablas.

El menú desplegado también puede contener esta opción de creación. Además todas las subpartes creadas a través de estos menús se asocian automáticamente, sin necesidad de especificarlo, al modelo desde el que se crearon.

Agregar una línea

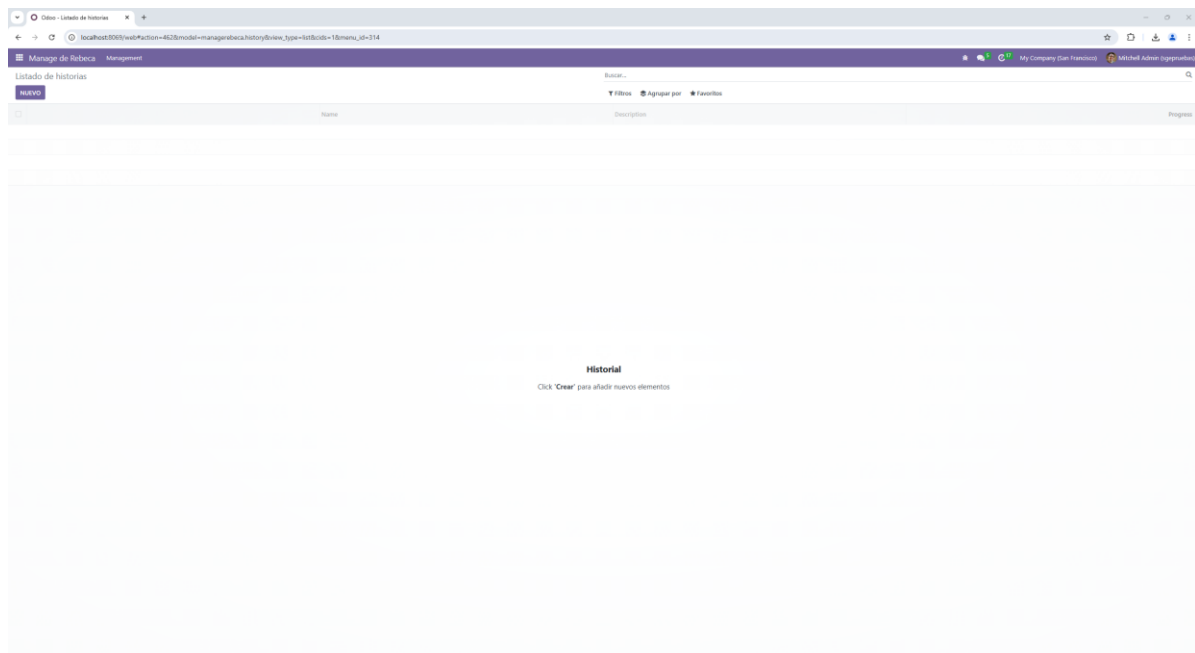
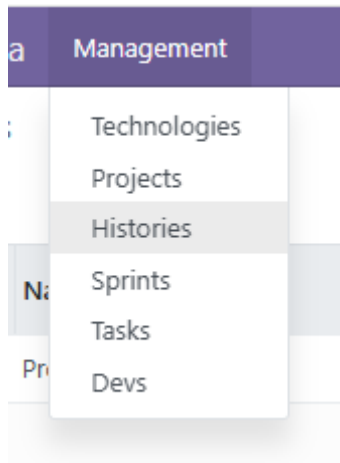


Este es el proyecto que acabamos de crear, el cual también se puede marcar como favorito.

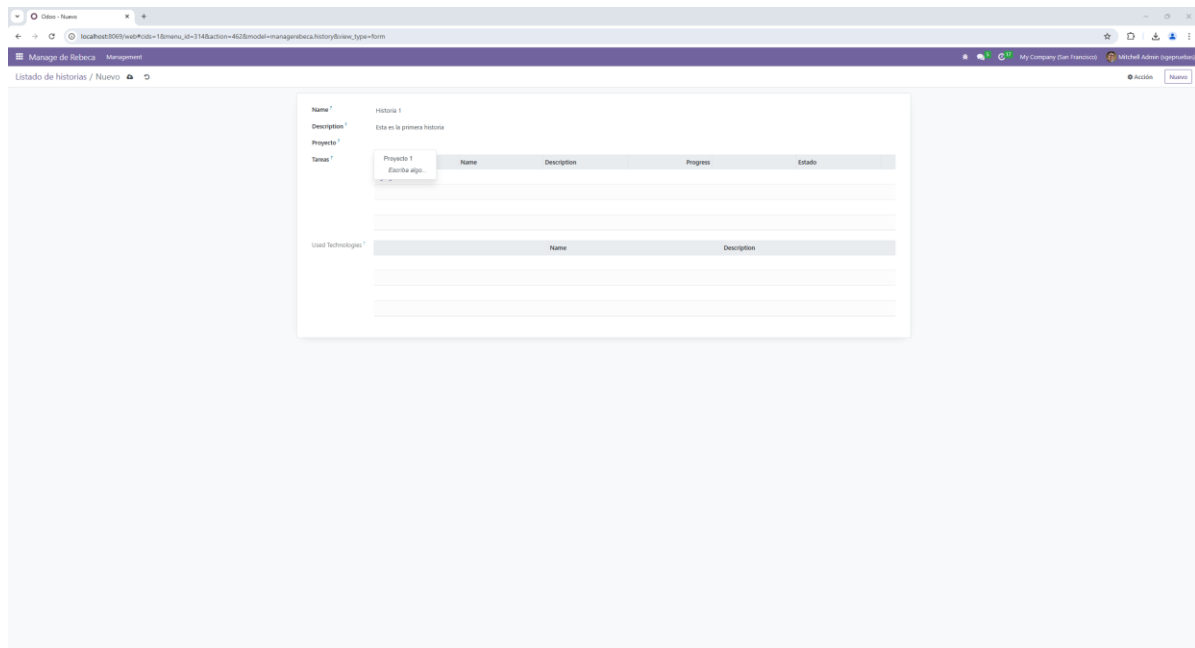


	Name	Description	Progress
☆	Proyecto 1	Este es el primer proyecto	0%

Nos dirigimos de nuevo al menú Management y esta vez pulsamos Histories y repetimos el mismo proceso, crearemos varias historias.

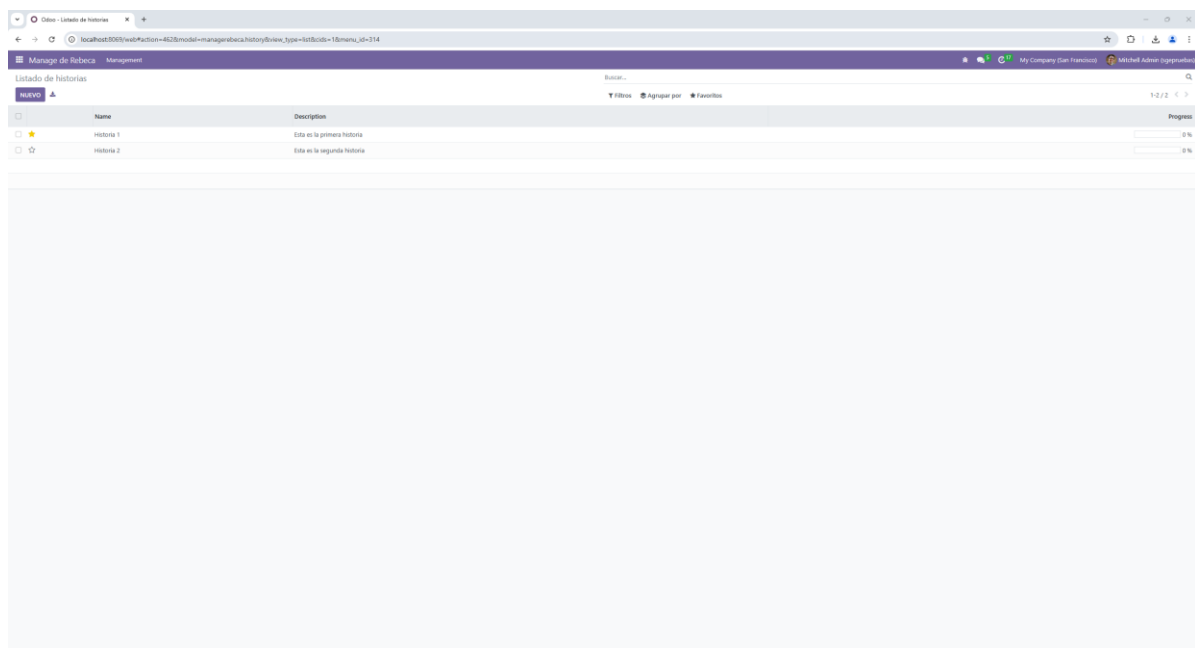


Como historias es un modelo que tiene relación con proyectos podemos asignar la historia que estamos creando al proyecto que acabamos de hacer mediante un desplegable que nos muestra todas las opciones disponibles.



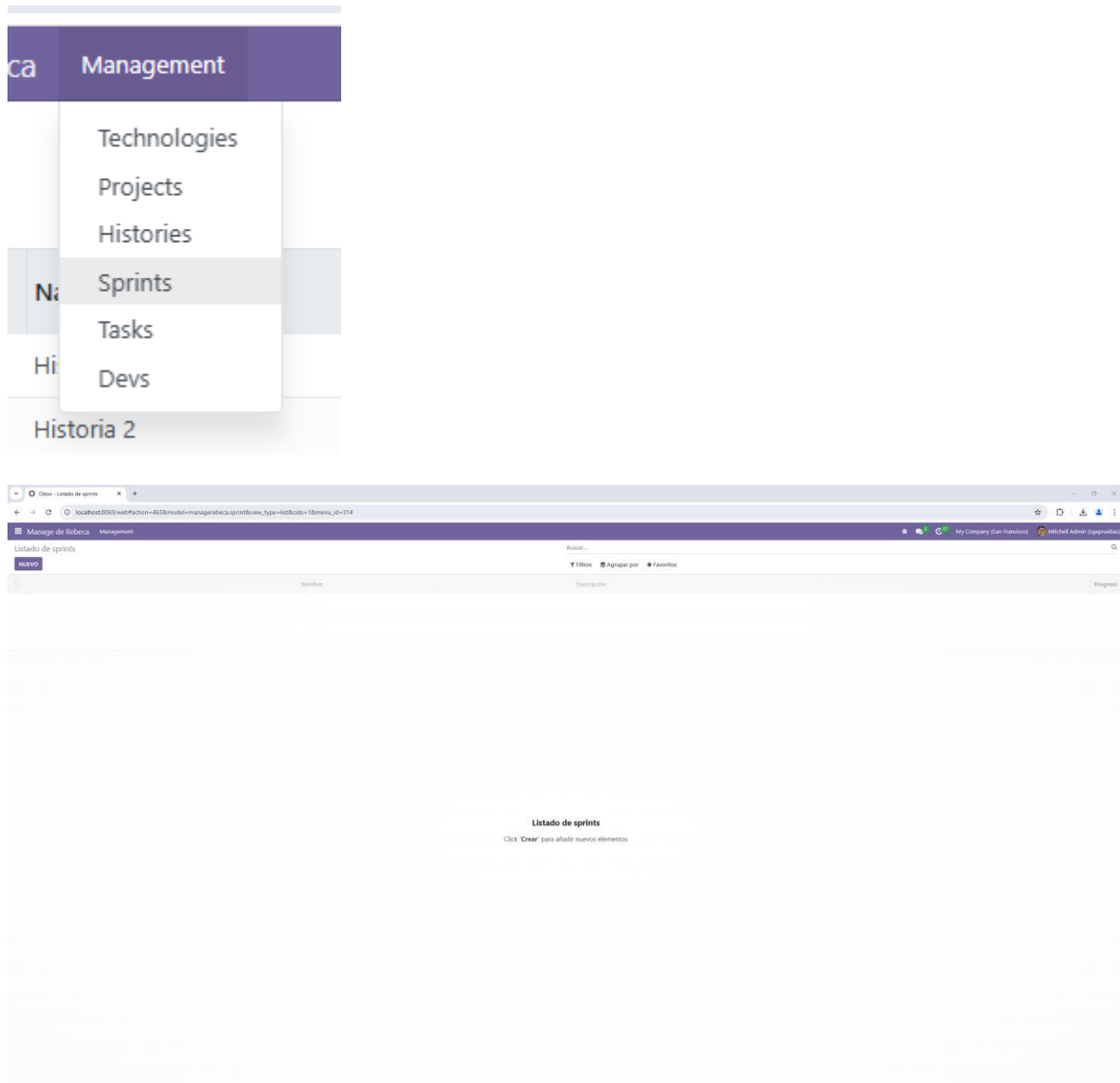
Nombre: Historia 1
 Descripción: Esta es la primera historia
 Proyecto: Proyecto 1
 Tareas: Escribe algo...

Name	Description

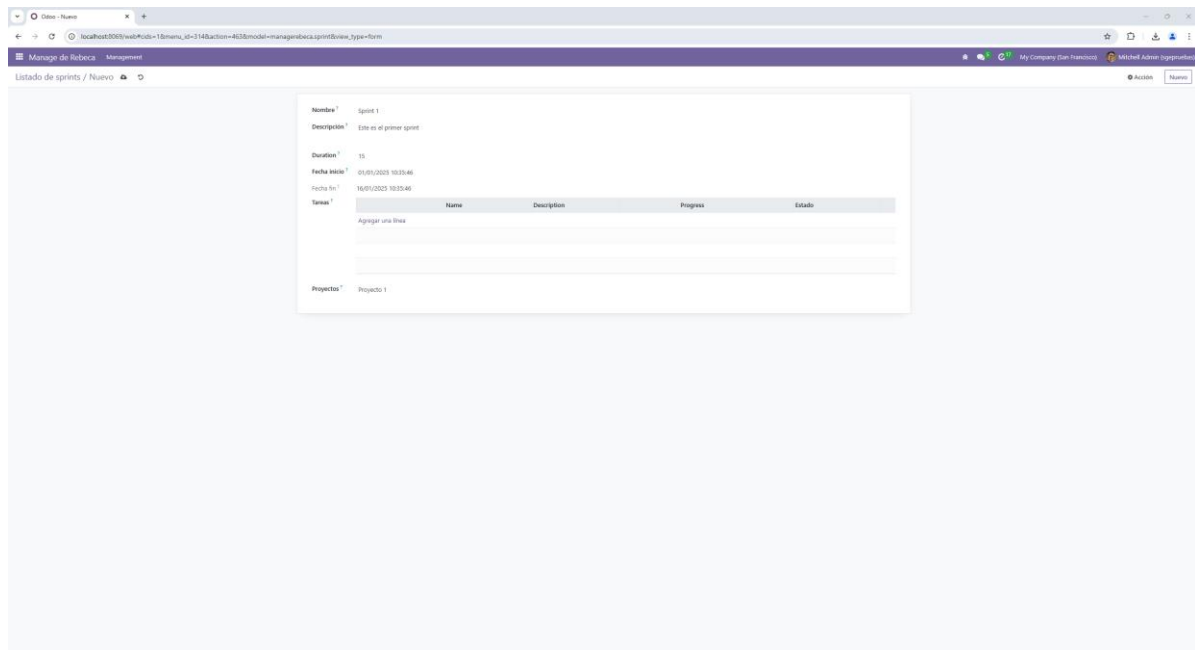


	Name	Description	Progress
<input type="checkbox"/>	Historia 1	Esta es la primera historia	0 %
<input type="checkbox"/>	Historia 2	Esta es la segunda historia	0 %

Ahora haremos lo mismo para los sprints, ya que estos también se relacionan directamente con el proyecto.

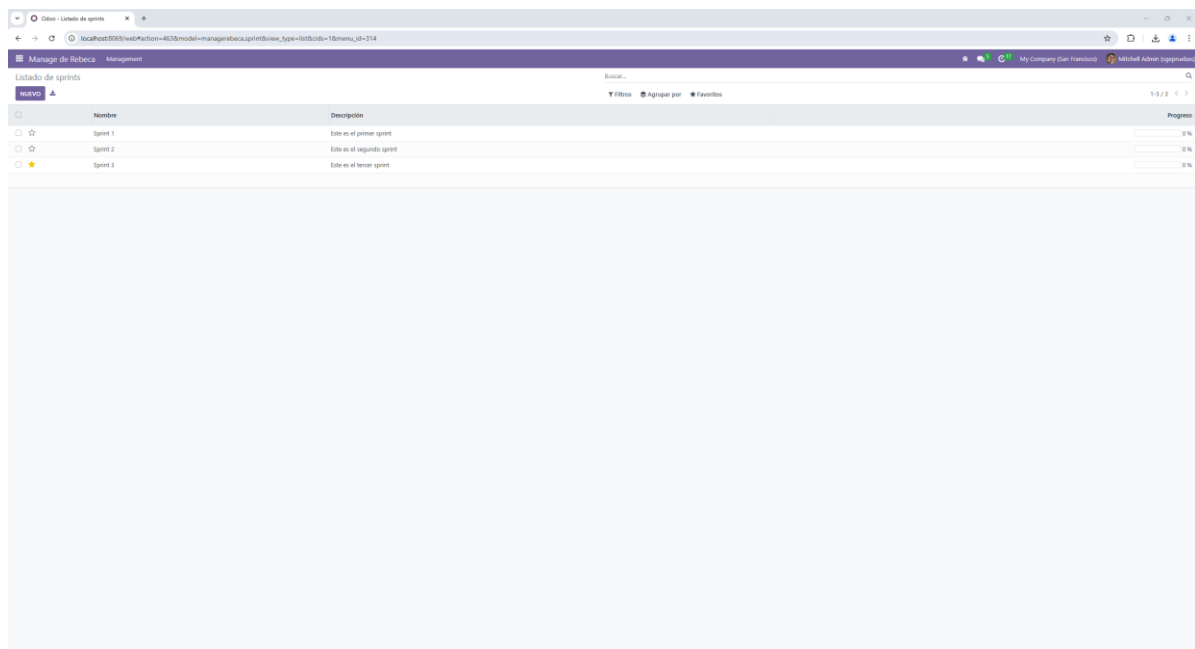


En los sprints la fecha de finalización se autocompleta añadiendo a la fecha de inicio la duración especificada que por defecto es 15.



The screenshot shows the 'New Sprint' form in the Odoo 'Manage de Retos' application. The form includes the following fields:

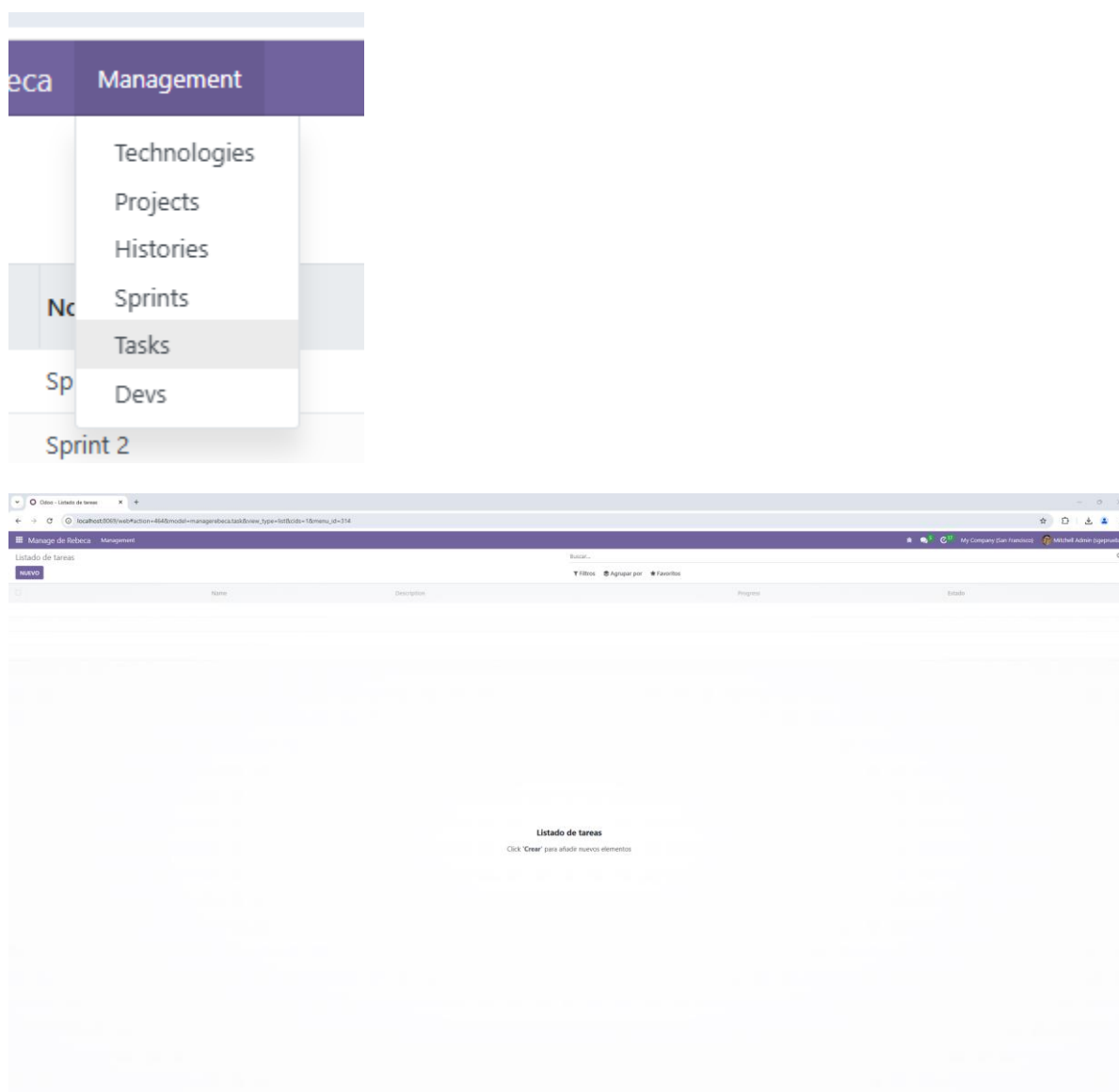
- Nombre**: Sprint 1
- Descripción**: Este es el primer sprint
- Duration**: 15
- Fecha inicio**: 01/01/2021 10:35:46
- Fecha fin**: 16/01/2021 10:35:46
- Tareas**: A table with columns: Name, Description, Progress, Estado. It contains one row: 'Agregar una linea'.
- Proyectos**: Proyecto 1



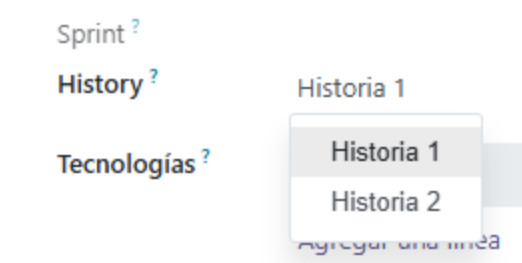
The screenshot shows the 'Listado de sprints' table in the Odoo 'Manage de Retos' application. The table has the following columns: Nombre, Descripción, and Progress. It contains three rows:

	Nombre	Descripción	Progress
<input type="checkbox"/>	Sprint 1	Este es el primer sprint	0 %
<input type="checkbox"/>	Sprint 2	Este es el segundo sprint	0 %
<input type="checkbox"/>	Sprint 3	Este es el tercer sprint	0 %

Por último crearemos las tareas. Accedemos de nuevo al menú Management y vamos al apartado Tasks.

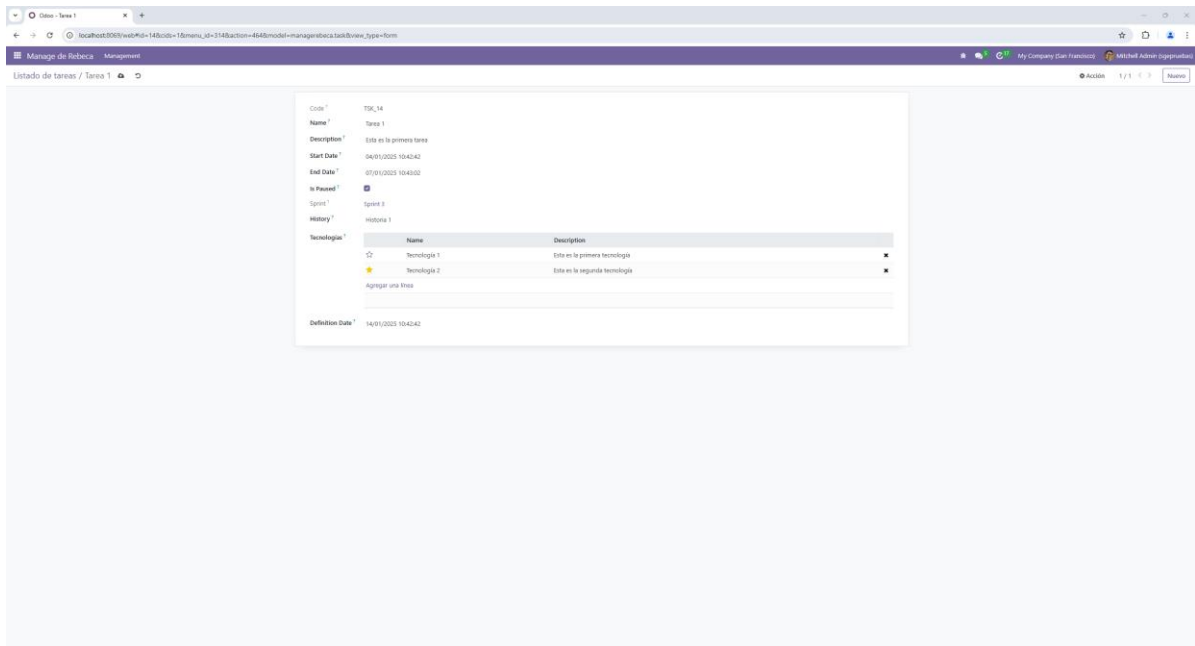
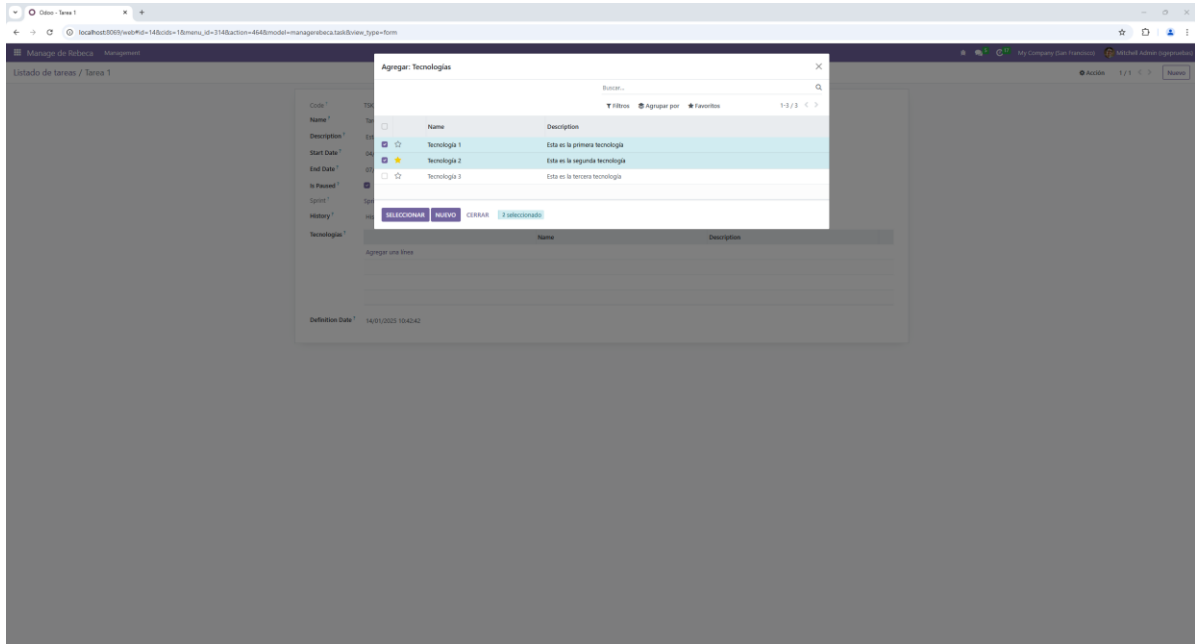


Podemos asignar la tarea a la historia que queramos pero no podemos escoger el sprint. Este se definirá automáticamente dependiendo de la fecha de finalización del sprint, la tarea se asignará al sprint que más lejana fecha de finalización tenga y que no sea anterior a la fecha actual.



Para agregar las tecnologías que se vayan a utilizar en esta tarea se debe pulsar sobre Agregar una línea en la tabla. Esto desplegará un menú donde podemos ver las tecnologías ya creadas y seleccionarlás o crear nuevas. Una vez escogidas pulsamos SELECCIONAR.

Agregar una línea



Una vez creada la tarea podemos ver que el sprint ya se ha autocompletado.

Sprint ?

Sprint 3

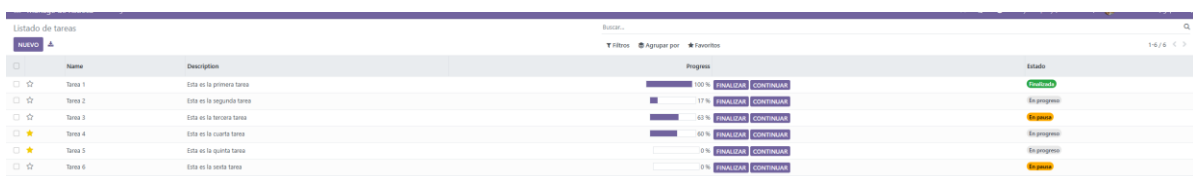
History ?

Historia 1

En la vista de las tareas podemos ver más información sobre estas. Para empezar tenemos el widget para marcar las tareas como favoritas.

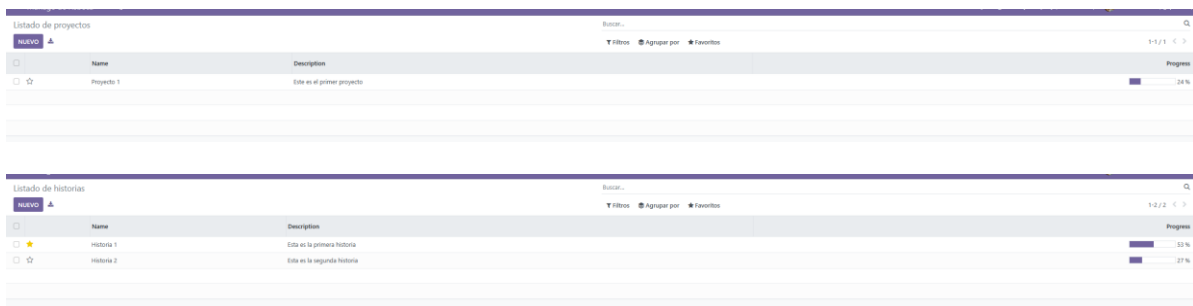
Luego tenemos una barra de progreso que muestra el porcentaje del tiempo transcurrido desde la creación de la tarea respecto al tiempo total para completarla. Si la tarea se completa antes del tiempo total se puede usar el botón FINALIZAR para marcar la tarea como finalizada y que el porcentaje de progreso se ponga en 100 automáticamente. Si finalizamos una tarea accidentalmente podemos usar el botón CONTINUAR para devolver el progreso al porcentaje del tiempo transcurrido.

Por último tenemos unas insignias que nos indican con colores y un breve texto el estado en que se encuentra la tarea. En color verde se definen las tareas finalizadas, es decir, aquellas cuyo porcentaje de progreso es 100. En gris se muestran las tareas que se encuentran en progreso actualmente y en amarillo aquellas que han sido pausadas.



	Nombre	Descripción	Progreso	Estado
<input type="checkbox"/>	Tarea 1	Esta es la primera tarea	100%	Finalizada
<input type="checkbox"/>	Tarea 2	Esta es la segunda tarea	17%	En progreso
<input type="checkbox"/>	Tarea 3	Esta es la tercera tarea	43%	En progreso
<input type="checkbox"/>	Tarea 4	Esta es la cuarta tarea	60%	En progreso
<input type="checkbox"/>	Tarea 5	Esta es la quinta tarea	0%	En progreso
<input type="checkbox"/>	Tarea 6	Esta es la sexta tarea	0%	En progreso

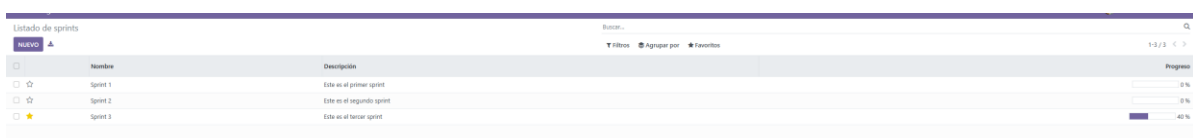
Una vez introducidos los datos de las tareas el porcentaje de progreso de los proyectos, las historias y los sprints se habrá recalculado en función de las tareas que tengan asignadas.



	Nombre	Descripción	Progreso
<input type="checkbox"/>	Proyecto 1	Este es el primer proyecto	24%

	Nombre	Descripción	Progreso
<input type="checkbox"/>	Historia 1	Esta es la primera historia	53%
<input type="checkbox"/>	Historia 2	Esta es la segunda historia	27%

En los sprints solo se han añadido tareas al sprint 3 porque el primero ya concluyó y el segundo tiene una fecha de finalización anterior a la del tercero.

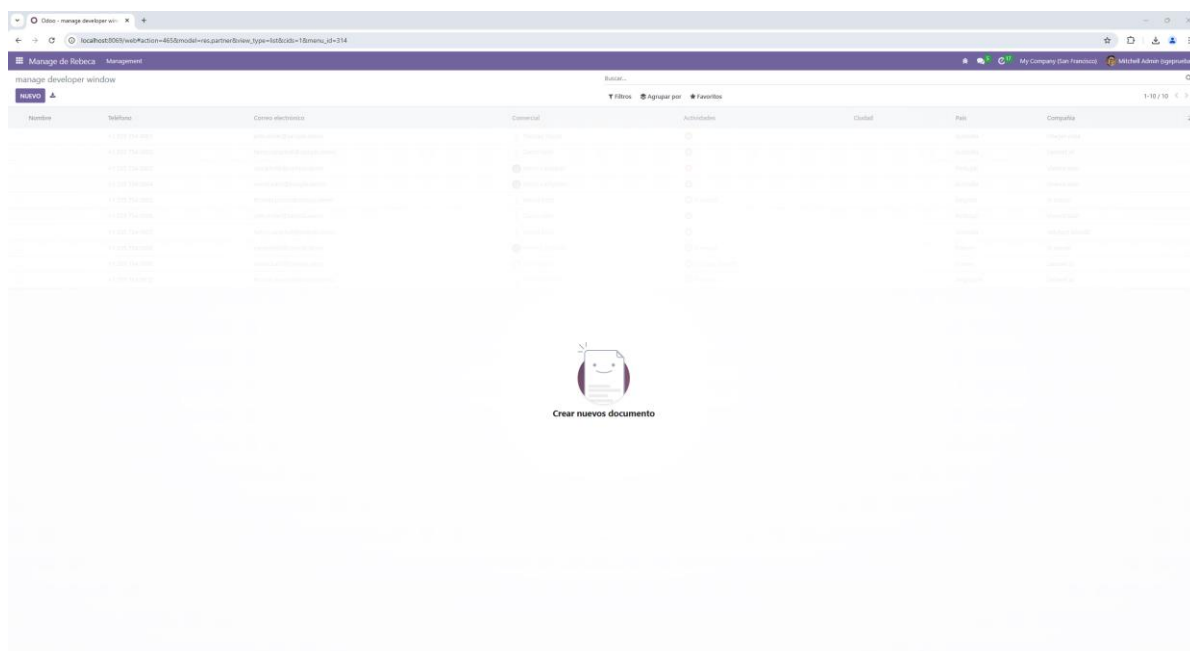


	Nombre	Descripción	Progreso
<input type="checkbox"/>	Sprint 1	Este es el primer sprint	0%
<input type="checkbox"/>	Sprint 2	Este es el segundo sprint	0%
<input type="checkbox"/>	Sprint 3	Este es el tercer sprint	40%

Por último veremos el apartado devs.



En este apartado toma como plantilla el módulo res.partner de Odoo para crear el contacto de los desarrolladores de nuestro módulo. Clicamos NUEVO y veremos el formulario de creación de contactos del módulo res.partner, creamos un nuevo desarrollador.



El desplegable nos permite elegir entre las empresas ya registradas en el módulo res.partner.

Nombre de la empresa...

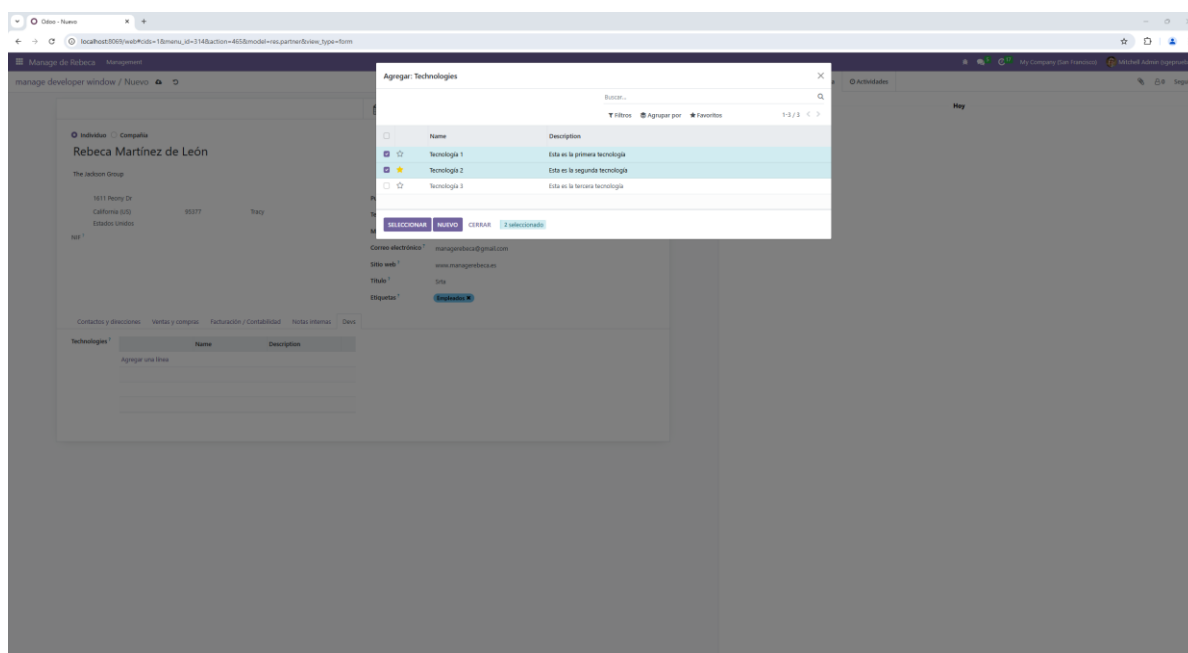
En la parte inferior del formulario se muestra el campo Devs que hemos añadido al formulario. En este apartado podemos seleccionar las tecnologías que utilizará este desarrollador.

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

Technologies ?

	Name	Description	
☆	Tecnología 1	Esta es la primera tecnología	✕
★	Tecnología 2	Esta es la segunda tecnología	✕

Agregar una línea



Ahora en la vista tree podemos ver los desarrolladores que vayamos creando.

Nombre	Teléfono	Correo electrónico	Comercial	Actividades	Ciudad	País	Compañía
<input type="checkbox"/> The Jackson Group, Rebeca Martínez de León	967654321	managerebeca@gmail.com			Tracy	Estados Unidos	

Conclusiones y posibles ampliaciones

En este proyecto hemos desarrollado una herramienta que cumple su objetivo de ayudar a la organización del trabajo mediante metodología Scrum. Para ello, se abordaron distintos temas como la integración de modelos relacionales o la creación de vistas dinámicas, todo ello enfocado a lograr una experiencia cómoda para el usuario.

Durante la creación del proyecto hemos tenido que aprender a utilizar herramientas nuevas como es Docker, la cual se utilizó para el despliegue y manejo de la base de datos del módulo. Además, la

creación de módulos en Odoo sigue un sistema de modelos y vistas que facilita el entendimiento del código y la posibilidad de desarrollar mejoras y ampliaciones en el futuro.

Algunas de estas ampliaciones podrían incluir la creación de una vista kanban. Esta permitiría la visualización de datos de una forma más cómoda. Las tarjetas podrían tener diferentes colores en función del estado de la tarea. También se podría cambiar el color de la barra de progreso dependiendo de cómo de avanzada esté esa parte concreta del proyecto.

Otra opción sería sustituir que la barra de progreso de las tareas avance en función del tiempo transcurrido y se base en el progreso que se ha hecho en cada tarea añadiendo una serie de puntos a completar en dicha tarea. Para completar se podría añadir un widget que, ahora sí, muestre cuántos días quedan para que termine la tarea. Este widget se podría usar en el resto de partes del proyecto como los sprints o las historias.

Bibliografía

<https://asana.com/es/resources/what-is-scrum>

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

<https://proyectosagiles.org/historia-de-scrum/>

<https://ilixum.com/evolucion-de-scrum/>

<https://entersol.com.mx/evolucion-sistemas-erp>

<https://wautechnologies.com/noticias/ranking-erp-mas-usados/>

<https://grupoaspasia.com/es/glosario/metodologia-scrum/>

<https://caroli.org/es/scrum-significado-aplicacion-conceptos-y-ejemplos/>

<https://code.visualstudio.com/>

https://www.odoo.com/es_ES

<https://www.docker.com/>

<https://ni.itc.services/blog/conoce-odoo-7/capitulo-1-descripcion-general-de-la-arquitectura-152>

https://www.odoo.com/es_ES/page/download

<https://ilimit.com/blog/metodologia-scrum/>

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

<https://konodoo.com/blog/konodoo-blog-de-tecnologia-1/post/40-widgets-de-odoo-para-crear-interfaces-llamativas-7>

<https://www.cybrosys.com/blog/widgets-list-in-odoo-14>

<https://www.odoo.com/documentation/18.0/applications/studio/fields.html>